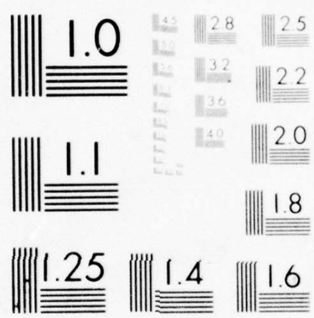
AD-A038 265 RESEARCH TRIANGLE INST RESEARCH TRIANGLE PARK N C OPE--ETC F/G 15/3
LOCAL EMERGENCY OPERATING SYSTEM - LEMOS.(U)
JUL 76 J W DUNN, R N HENDRY, R O LYDAY DAHC20-73-C-0253
UNCLASSIFIED RTI-44U-873 NL

1 OF 2
AD
A038265



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

RESEARCH TRIANGLE INSTITUTE

July 1976

DCPA Work Unit 4126I

Contract DAHC20-73-C-0253

Final Report 44U-873

LOCAL EMERGENCY OPERATING SYSTEM - LEMOS

by

R. N. Hendry
R. O. Lyday
J. W. Dunn

Prepared for:

Defense Civil Preparedness Agency
Department of Defense
Washington, D.C. 20301

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

RESEARCH TRIANGLE INSTITUTE
Post Office Box 12194
Research Triangle Park, North Carolina 27709

(6)

RESEARCH TRIANGLE INSTITUTE

Operations Analysis Division
Management Information Sciences Department
Post Office Box 12194
Research Triangle Park, North Carolina 27709

FINAL REPORT 44U-873

July 1976

LOCAL EMERGENCY OPERATING SYSTEM - LEMOS

by

R. N. Hendry, R. O. Lyday, and J. W. Dunn

for

DEFENSE CIVIL PREPAREDNESS AGENCY
Department of Defense
Washington, D.C. 20301

under

Contract DAHC20-73-C-0253 new
DCPA Work Unit 4126I

DCPA REVIEW NOTICE

See app. B.

FORWARD

The research reported herein covers the "design" phase of the countermeasures model as a part of a computer based procedure for the evaluation of local operating systems. This work is sponsored by the Defense Civil Preparedness Agency (DCPA) under Contract DAHC20-73-C-0253, Work Unit 4125J.

The authors express their indebtedness to Mr. Donald Hudson of the DCPA for his guidance during the study. The authors also express their appreciation to Mr. Edward L. Hill and to others in the Research Triangle Institute who supported this research.

ABSTRACT

The Local Emergency Operating System (LEMOS) is a multi-program model within a system of models which are designed as a part of a computer based system to evaluate local operating systems. The Defense Civil Preparedness Agency Computation Center is a co-developer with RTI in the development of ADS/LEMOS. Effort by RTI during the past year has centered around the development of the control and transportation submodels and at the same time has continued to improve the procedures within other submodels.

This report describes the essential features of the control and transportation models. In addition, a brief discussion is included covering the application of the ADS/LEMOS model to the local CD planning.

The report concludes that emphasis should shift now from design to development but with the specific objective of gaining support for continued evolution to an operational state.

Adequate developmental testing is recommended before any demonstrations of the planning role are undertaken.

TABLE OF CONTENTS

LIST OF FIGURES

DETACHABLE SUMMARY

The Local Emergency Operating System (LEMOS) is a series of interrelated computer programs which operate as a part of a larger system of programs and manual procedures designed to "Test and Evaluate Local Operating Systems" (TELOS).

Simulation of local operations is practicable by the use of computer-based models provided the user avoids too much detail which renders the simulation time-consuming, and, therefore, expensive. The authors believe that, while the current TELOS design is seemingly complex, it has, in fact, avoided this pitfall and achieved a suitable balance between too little and too much detail. The level of detail adopted is thought to give enough realism or believability and, yet, not require endless data processing. This report is the latest in a series of reports describing the evolution of LEMOS. The most recent effort, culminating in this report, includes the control interface development and the transportation model development described in Section II and III, respectively. While the interface between ADS and LEMOS is believed to be satisfactory, it has not been tested to verify its performance.

Command and control over local civil defense operations would be achieved, in the real world, through an information system utilizing communication networks connecting command and control centers. In the LEMOS model command and control is achieved through the definition of policies, priorities, and prohibitions in coded form within the control file. The absence of a communication submodel as is the present case is tantamount to the assumption that communications are "perfect". Since the prototype LEMOS model does not include communications, the command and control model is defined as a control file (containing discrete values for selected policies, priorities, and prohibitions) and the Generalized Executive Control (GENEC) which activates the scenario.

The GENEC system enables a tape of executable modules (load and go modules of the major sub-programs in the LEMOS system plus other modules, as required) to be executed repeatedly in any sequence desired and controlled by a second parameter tape. The system specifically iterates between the countermeasures and damage assessment models, continuing through a number of time periods under control of GENEC. Thus, the basic scenario may be embodied in GENEC by the controls residing in the Control File accessed through it. Manual evaluation of system outputs is contemplated at this time. A mechanized output data processor may be needed to analyze the relatively large output from both models with respect to the particular

role of TELOS. On the basis of this evaluation, controls for further scenarios may be determined and implemented through the use of GENEC. Throughout the course of the simulation, reports may be generated to measure the status of the system under test as an aid in the evaluation of local emergency operations.

The major effort during this contract period has been to develop the transportation submodel or, more accurately, the "shortest path algorithm". This submodel is described in considerable detail and is viewed as a significant addition to LEMOS.

If LEMOS is to have a practical role in local CD planning, a plan by which this role can be attained is essential at this time. It may be fairly stated that the Research phase of RDT and E is nearing completion. The Development, Test, and Evaluation phases should begin immediately. During these phases, selected Federal, State, and Local planners should be enlisted to participate in evolving and conducting them. It is absolutely imperative that they have an important impact on the final configuration of the simulation system before it is used operationally to improve the local CD planning function.

DCPA is strongly urged to develop appropriate multi-year program plans to conduct Development, Test, and Evaluation phases of the TELOS system with participation by Federal, State, and Local planning personnel. Special emphasis should be placed during DT and E phases on the use of Case Study Areas. These plans should include the addition of the following elements during the Development phase:

Local plans Pre-processor,
Interactive Control Procedures,
Fire Spread Model (developed by not interfaced with ADS),
Special Resource Damage Submodel,
Communications Submodel,
Utilities Network Submodel,
Medical/Epidemiology Submodel.

LOCAL EMERGENCY OPERATIONS SYTEM (LEMOS)

I. INTRODUCTION

A. General

The Local Emergency Operating System (LEMOS) is a series of interrelated computer programs which operate as a part of a larger system of programs and manual procedures designed to Test and Evaluate Local Operating Systems (TELOS). Figure 1 portrays the general configuration of the larger system. It can be seen that the executive control, damage assessment, and countermeasure segments of that system have a particularly close relationship to each other. This report is the latest in a series of reports describing the evolution of LEMOS. The most recent effort, culminating in this report, includes the control interface development described in Section II and the transportation model development described in Section II and III, respectively. While the interface between ADS and LEMOS is believed to be satisfactory, it has not been tested to verify its performance. Future work should concentrate on obtaining successful performance tests between the various segments identified in Figure 1 and on performing case studies employing TELOS as a planning tool.

First, in order to use TELOS as a planning tool, appropriate procedures must be developed to translate existing planning documents into machine readable inputs and to convert computer outputs into suitable planning documents. An approach to this problem is described briefly in the subsection entitled "Local CD Planning."

Second, assuming that local planning methodology can be made compatible with the TELOS concept, the local planners must understand the simulation characteristics of the model and know how to use it to produce probable outcomes that they believe represent their real-world situation. This methodology is discussed under subsection C.

And third, the local planners need to learn to interpret and evaluate simulation outcomes in terms that allow them to make decisions regarding the

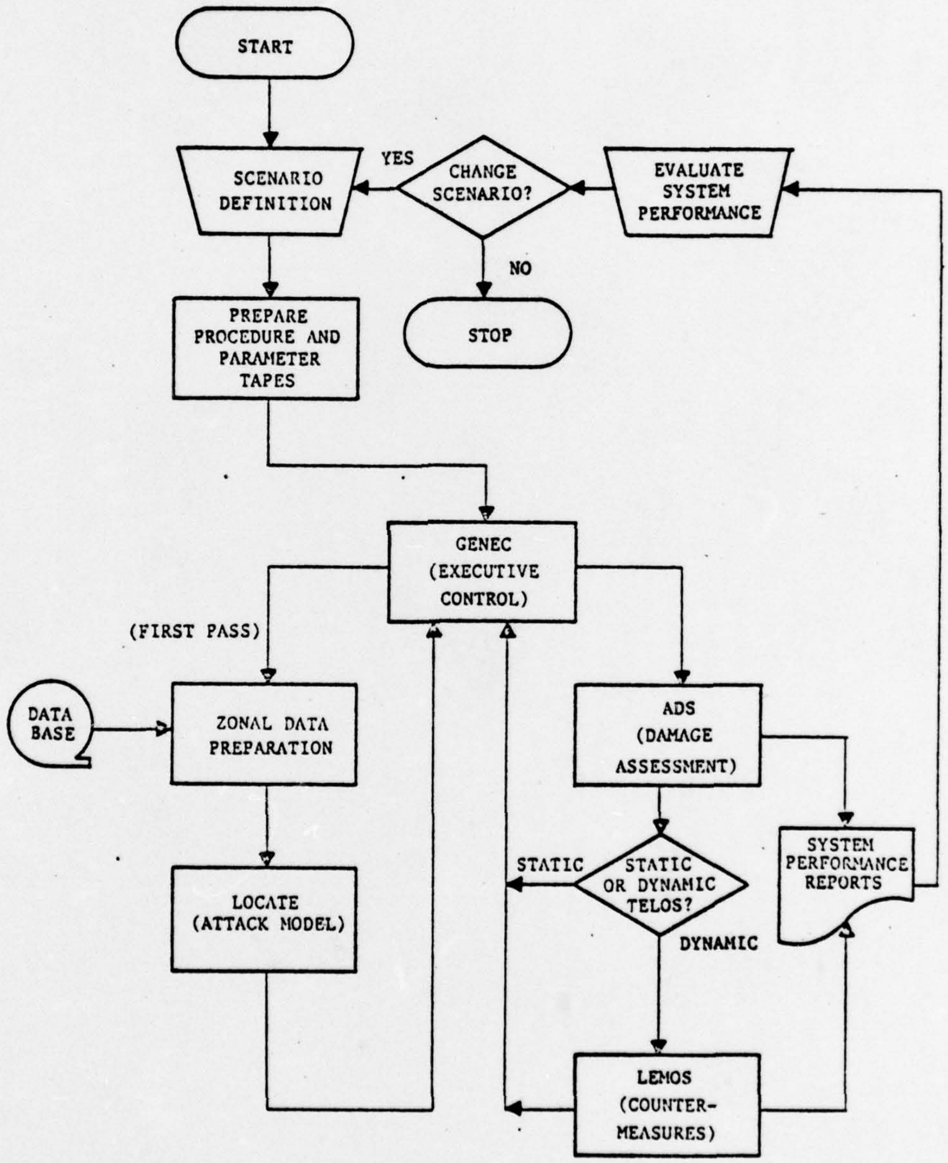


Figure 1. Overall TELOS Description

status of local Civil Defense represented by the simulation to achieve the most effective use of available or anticipated resources. The last subsection in this section is addressed to the evaluation function. However, it does not presume to cover this area adequately as, the authors believe, this function must be developed in close cooperation with local officials. For without their input and sanction, TELOS will not achieve success as a planning tool.

B. Local CD Planning

Planning is the primary responsibility of management. In fact, the local CD Director must be considered to be the chief planner and must devote a very significant part of his time to discharging this responsibility. Not only should he have written plans which give qualitative evidence that he is fulfilling this duty, but, they must be susceptable to quantitative evaluation to provide adequate assurances that they represent the best available plans. In short, higher authorities need objective methods for the evaluation of local CD planning.

In the era of detenté and SALT agreements, the most cost-effective CD activity that can be employed is planning. It is the least costly measure with the highest possible payoff and should precede all other preparedness activities.

Current Federal guidance provides direct assistance to the local planner; however, it demands little from him to give the necessary assurances that local planning will make the best use of available resources. Nor does it suggest reasonable utilization of additional resources should they become available.

The TELOS system of computer programs offers a means for measuring, through simulation, the ability of local plans to meet nuclear disaster scenarios deemed probable for those areas to which they apply.

At present, TELOS has not been finalized or tested in the planning role. Furthermore, it should not be completed until local planners can participate in finalizing the interface procedures and pass judgment on its merits as a planning

tool. TELOS must not be employed to denegate the adequacy of the current level of local planning. In general, most planners are painfully aware of the shortcomings of their plans. It should be used to stimulate better planning by revealing operational inadequacies and reporting on resource needs. The basic method adopted in TELOS is one of simulating local operations with finite resources.

C. Simulation of Local Operations

Simulation of local operations is practicable by the use of computer-based models provided the user avoids too much detail which renders the simulation time-consuming, and, therefore, expensive. The authors believe that, while the current TELOS design is seemingly complex, it has, in fact, avoided this pitfall and achieved a suitable balance between too little and too much detail. The transportation submodel described in Section III typifies this belief. The network could have been more or less detailed. The level of detail adopted is thought to give enough realism or believability and, yet, not require endless data processing.

The development of a believable simulation will depend on the effective integration of the local planner's ideas into the simulation process. They are the ones who must adopt the concept, if TELOS is to succeed as a planning tool, and they are less likely to be advocates of the system, if they are not participants in its final development. The TELOS submodels developed to date represent the main frame of the potential structure. The elements yet to be developed are the elements upon which the planner will have the highest impact. These elements may be loosely characterized as the input and output data processors. Assuming that appropriate input and output interfaces are developed with local planners and their acceptance of the system attained, planning through simulation can proceed.

Initially, a local area may be described by its physical resources, its countermeasure plans, and the various probabilistic attack scenarios. Simulation runs may be made to determine outcomes. Information gained during these runs may suggest improvements in the countermeasure plans. Revised plans may be rerun to verify the predictions. Quantitive measures gained during the process provide means for optimumizing local losses and resource allocations.

The models are sufficiently flexible to accommodate almost any local configuration and may be run as may times as time and effort permit to achieve a balanced set of plans uniquely applicable to that area.

D. Planning Evaluation

Since each local CD planner is responsible for his plans, TELOS cannot assume the evaluative role for him. Therefore, the outcome interface must contain several alternative means for evaluating simulation outcomes as they may be influenced by local plans. The planner should have not only the choice from among these alternatives but should be able to "tune" the selected method to meet his objectives wherever practicable. Measures for normalizing outcome measures is especially important.

Previous reports have described readiness and benefit measures which TELOS may produce as a result of the simulation of local operations. Additional measures may be generated if the local planner considers them essential to the proper evaluation of his plans.

Two elements of local planning are considered in the next two sections of this report. They are important considerations in any local planning simulation and are described as a part of the development of the ADS/LEMOS subsystem.

II. COMMAND AND CONTROL MODEL

A. Introduction

Command and control over local civil defense operations would be achieved, in the real world, through an information system utilizing communication networks connecting command and control centers. In the LEMOS model with a communication (COM) submodel, command and control is achieved through the definition of policies, priorities, and prohibitions in coded form within the control file. The absence of a communication submodel is tantamount to the assumption that communications are "perfect". In this view, the presence of a COM submodel permits the evaluation of a delay between an event and the generation of data (information), between its generation (transmission) and reception, or between its reception and decision-making. Loss of information is equivalent to an infinite delay.

Since the prototype LEMOS model does not include communications, the command and contol model is defined as a control file (containing discrete values for selected policies, priorities, and prohibitions) and the Generalized Executive Control (GENEC) which activates the scenario.

B. Generalized Executive Control

The Generalized Executive Control (GENEC) system represented by Figure 2 was developed by the Defense Civil Preparedness Agency Computer Center (DCPACC) and was designed to run on the CDC 3600 at DCPACC under the SCOPE[1] operating system. The CDC 3600 and SCOPE system is a single program execution computer system with no inherent provisions for sequentially repeating a set of programs. The ADS and LEMOS series of programs should be executed as a system and not as separate runs due to the interactive and iterative nature of the process. The GENEC system was specifically developed to accomplish this task. Its operation is shown diagrammatically in Figure 2 and in an alternate system concept, would be

---

[1] software system operating on the CDC-3600.

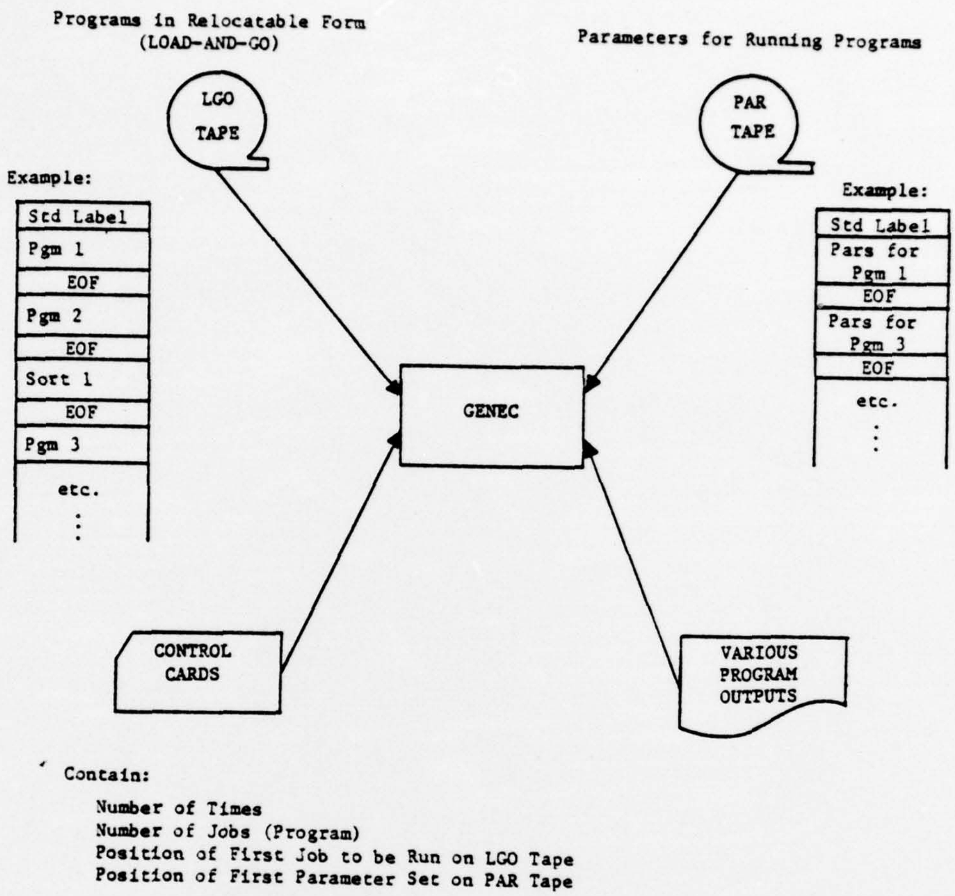


Figure 2. Conceptual Model of GENEC

interactive and capable of being dynamically modified as it executes successive passes.

GENEC is initiated by control card input, which tells GENEC the number of times that the job stream is to be repeated, the number of jobs in the stream, the position of the first job on the program tape and the position of the first set of parameters on the parameters tape. In addition, GENEC will be able to pass approximately five words of data from one routine to the next routine executed by the memory mapping technique.

C. ADS/LEMOS/GENEC Operation

The overall flow of the TELOS system is under the control of manual intervention and GENEC as outlined in Figure 2.

The GENEC system enables a tape of executable modules (load and go modules of the major sub-programs in the LEMOS system plus other modules, as required, from ADS) to be executed in any sequence and executed repeatedly, if desired, as controlled by a second control or parameter tape.

The system specifically iterates between the countermeasures and damage assessment models, continuing through a number of time periods under control of GENEC. Thus, the basic scenario may be embodied by GENEC and controls residing in it and in the Control File accessed through it. Manual evaluation of system outputs is contemplated at this time. A mechanized output data processor may be needed to analyze the relatively large output from both models with respect to the input and the particular role of TELOS. On the basis of this evaluation, controls for further iterations may be determined and implemented through the use of GENEC, and a new cycle begins. Throughout the course of the simulation, reports may be generated to measure the status of the system under test as an aid in the evaluation of local emergency operations.

The main data linkage between all of the various LEMOS submodels, as shown in Figure 3, is the Control File, although some 13 file types are used together or individually as transitions between programs. The Control File is, however, only modified by the manual intervention exterbak to ADS/LEMOS.

D. GENEC/Control File Interaction

As a part of each sub-program in the LEMOS system, there is a set of parameters that determine the type of run, type of input, and type of output that a module is to provide.

The "essential" parameters are used to control the functions of each of the

(Being Drafted)

Figure 3. Local Operating System Overview

programs in LEMOS. There are three "essential" parameters which are input to each program, called RUN-SWITCH, TST-SWITCH, and PRNT-SWITCH. The first of these, RUN-SWITCH, is generally used to select which major functions a program should perform. The second parameter, TST-SWITCH, is generally used for aid in debugging or testing a program. The last parameter, PRNT-SWITCH, is generally used to select printout options within a program, e.g., which table(s) to list.

A GENEC modification provides a way to pass a few parameters from one module to another (five 48 bit words) and, therefore, the execution of the full LEMOS system can be previous history dependent as well as having its basic execution being dependent on the planned scenario.

The control file and procedures to access it are contained in a copy library which is invoked at the time of program compilation. This procedure assures uniformity in the application of policies, priorities, and prohibitions to all procedures.

All programs in the LEMOS model, including the transportation model described in the next section, have access to these control procedures.

III. TRANSPORTATION MODEL

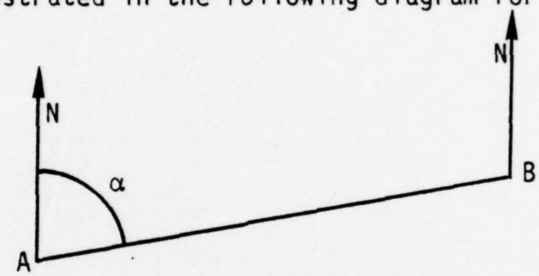
A. General

The major effort during this contract period has been to develop the transportation submodel or, more accurately, the "shortest path algorithm". This submodel is described in considerable detail in Appendix B and discussed in the following sub-sections. It is viewed as a significant addition to LEMOS and, indeed, to the entire ADS/LEMOS system.

B. Network Description

Freeways and major arteries are the basis of the current development of the transportation network. Residential and feeder streets are not included as they unnecessarily expand the network to unmanageable proportions. Intersections of highways and arteries define nodes in the network. Additional nodes could be added, if desired, at non-intersection points. The segment between two nodes is a link in the network. Nodes are identified by unique numbers. Connectivity of Unit Areas is determined by the occurrence of equal node numbers. Links are identified by numbers and names. Link identification is not involved computationally in the current computerized transportation sub-model; it is used only to aid in creating the links file data base. The transportation network is described by the Links File. The data in this file is used to create the matrix of distances between directly linked nodes in a unit area network. Each record of this file defines a link, primarily, in terms of its two terminal nodes, the distance in miles between the nodes, and the type of link (i.e., one-way or both ways). The two nodes are identified as a backward node and a forward node. The distinction is not critical except in the case of one-way streets and when the distance from one node to the other differs from the converse in which two different links records are prepared. The link type is used to distinguish between one-way and two-way links. It could also be used to describe other

transportation networks, i.e., railroads or waterways. However, our present effort is limited to highway networks. In earlier work, the forward node was defined as follows: if the angle measured from due North taken about one of the two nodes to the straight line connecting the nodes is greater than 180 degrees, that node is the forward node. From this it follows that the other node is the backward node and, if the angle is less than 180 degrees, the converse holds. The definition is illustrated in the following diagram for two nodes A and B and



illustrates the definition of forward and backward nodes. The indicated angle, α, about node A is less than 180 degrees, therefore, A is the backward node and B is the forward node.

In the current version, the definition is retained, but the emphasis is on the distance and direction of travel between the two nodes and the associated link codes. The table in Appendix B, Input-Output Description Section, summarizes the codes used in the links file.

In the present version of the computerized Transportation Submodel, no test is made on the first character of the link code. Thus, only major arteries (i.e., roadways) should be included in the Links File. If other types are included, it should be borne in mind that a constant speed of 45 mph is used to convert distances to travel time, as coded in the Links File and adjusted by the Basic Operating Situation (BOS) in the Resource File.

The table in Appendix B, Input-Output Description, describes the content of the records in the Links File.

In the previous work Ref 1 , four levels of networks coinciding with the organizational structure were devised because the estimated number of links in the anticipated networks exceeded computer capacity when the standard procedure of storing the distance matrix in computer core was used. The current computerized version, using out-of-core storage of the matrix, can deal with relatively large networks. The reasons for retaining the network-level and node-level codes in the Links File data are to provide visual checks in data preparation and to provide the means necessary to view the transportation problem as sector to sector instead of unit area to unit area, if this should ever become desirable in the future. Essentially, the only changes necessary would be to remove the '999' divider cards in the Links File between those unit areas to be agregated and to change the coding in tests of the network identification number to test the value of the variable corresponding to the appropriate level. Linkage between higher level networks would be determined by equality of node identification numbers having a node-level code appropriate for the network level.

Node-level codes and their meanings are summrized as follows:

Node Level Code	Meaning
1	Interior to unit area; not shared with other areas
2	Shared between unit areas
3	Shared between sectors
4	Shared between groups
5	Shared between EOC's
6	On the boundary of zones.

A node with a level code of 4 is shared by (at least) two groups, i.e., it is on the boundary between two groups. The node may occur as a boundary node between some sectors, constituents of those groups, and some unit areas,

constituents of those sectors. The value assigned the node-level code should correspond to the highest level occupied by the node. This, however, is not a requirement for the correct execution of the current version of the computer program.

C. Minimum Travel Time (Path) Between Unit Areas

The computerized Transportation Submodel computes the paths of minimum travel time between all unit areas and creates the TVL-REC file. The problem is defined and solved in terms of distances, and the conversion from distance to travel time is made by assuming a speed of 45 miles per hour. The input data to the computer model consist of two files, the Links File describing each unit area network in its initial state, and a Control File, which identifies the numbers of the unit area networks selected for processing. The computer program has been written to accommodate unit area networks containing a maximum of 75 nodes each and a network comprising a maximum of 400 unit areas. It has been tested on a network of 8 unit areas containing from 6 to 26 nodes.

There are two major steps in the model. The first is a "one-for-all" step that computes shortest paths between all nodes in a unit area network and the distances between directly linked unit area networks. This step used the Link and Control Files previously mentioned. The second step uses the Basic Operating Situation of each unit area to alter these distances and, then, to compute the shortest paths between all unit areas. The second step can be repeated as often as necessary as BOS changes occur over time in a given scenario.

Both steps use Floyd's Algorithm to compute the shortest paths between all nodes. The major difference between the two steps in the application of the algorithm is that in the first step (distance between nodes in a unit area) the entire distance matrix is stored in computer core memory while the second step takes advantage of the fact that the algorithm operates upon only two rows of the distance matrix at a time. Thus, only two rows of the distance matrix need be in

core at any time. The remainder of the matrix is stored on tape or disk. Substitution of tape or disk for computer core memory permits the direct processing of extremely large matrices, i.e., large numbers of unit areas, which, if stored in core, would exceed the capacity of the computer to be used. This ability to compute directly the shortest paths between unit areas obviates resorting to the previously described hierarchical scheme in which unit areas were aggregated into sectors, sectors into groups, and so on, where shortest paths were computed between the constituents of each hierarchical level.

Two unit area networks are directly-linked if they share one or more nodes in common. Such nodes are called boundary nodes. The distance between two directly-linked unit area networks from A to B is defined as the average distance taken over shortest paths from all nodes in A to all nodes in B via the boundary nodes between A and B. If A and B are not directly-linked, the distance between them is set at a large number for computational purposes. The distance from A to B is not necessarily equal to the distance from B to A because of, for example, the possible occurrence of one-way streets. In application, two distance matrices are maintained. In the first matrix, each element represents the average distance (over shortest paths) from all nodes in a given unit area to the boundary nodes it shares with some directly-linked unit area. The elements of the second matrix are the average distances from the common boundary nodes to all other nodes in the other unit area. Each element of the final matrix of distances between directly-linked unit areas is the sum of the corresponding elements of these two matrices divided by the corresponding number of common boundary nodes.

In a time scenario, damage is inflicted upon (or removed from) a unit area resulting in a change in travel time within the area. The status of the unit area is not determined to any finer resolution down to the level of individual blocks. Thus, the damage is assumed to be distributed uniformly over the unit area. All initial state shortest paths within a unit area, in effect, are increased by a

factor greater than 1, if the state of the area has deteriorated, or decreased by a factor less than 1, if conditions have improved. This apparent change in distance results in an observed change in travel time.

Values of BOS range from 1 to 9 and indicate the status of the environment in a unit area. The value includes the effects of debris, fires, and radiation. Somewhat arbitrarily, the increase in travel times associated with an increase in BOS is functionally represented by

$$f = 2^{(BOS-1)} .$$

Values of BOS, their meaning, and the associated increase in travel times are summarized as follows:

BOS Values	Meaning* Debris	Fires	Radio-activity	Increase in Travel Time
1	N	N	N	X 1
2	N-M	M	N	X 2
3	S	M	N	X 4
4	N	S	M	X 8
5	M	S	M	X 16
6	S	S	M	X 32
7	N-S	N	S	X 64
8	N-S	M	S	X 128
9	N-S	S	S	X 256

* N = Negligible   M = Moderate   S = Severe

The intent is to increase the travel time in an area with a BOS of 9 so that the shortest paths algorithm is unlikely to select that area as an intermediate node. In other words, such areas are to be avoided. Alternative weighting systems may be used if the results warrant changing the method adopted herein.

While it is not necessary to recompute the shortest paths within a unit area, since all distances are multiplied by the same factor, it is necessary to recompute the shortest paths between unit areas. To illustrate, consider the case where a single unit area, Q, has sustained damage, i.e. an increase in BOS, in a network comprising a total of n unit areas. Basically, two situations can occur.

The first situation exists where Q is the origin or the destination in the shortest path between two unit areas. These shortest paths, containing intermediate unit areas which have not sustained a change in BOS, are only changed by an additive amount resulting from the increase sustained by Q, which is at the end of the paths. Thus, it is not necessary to employ an algorithm to resolve the shortest paths problem in this situation.

The second situation involves Q as an intermediate area in the shortest path between two other areas. In this situation, a shorter path may result by using some area other than Q as an intermediate area. Thus, it is necessary to resolve the shortest paths problem in this situation. A description of the various techniques considered for application in this situation follows.

In addition to Floyd's algorithm (which solves for all shortest paths from all nodes in a network to either one or all other nodes), another efficient method is Dijkstra's algorithm, which solves for either the shortest path between two specified nodes or the shortest paths from a specified origin to all destinations. Identification of those shortest paths, where Q is an intermediate node, would then provide a list of origin/destination-specific paths to be resolved using Dijkstra's algorithm. The disadvantage to this approach lies in the storage and retrieval by rows of not only the matrix of shortest paths but also the policy matrix used initially by Floyd's algorithm to solve for all shortest paths between all nodes. Essentially, the disadvantage is that an excessive amount of computer input/output time would be consumed in reading from storage various rows of these two matrices in a random, non-predictable order. The basis for this conclusion

will be discussed in greater detail. However, as a consequence of it, the problem can be resolved by a re-application of Floyd's algorithm in approximately the same amount of computing time required for an application of Dijkstra's procedure with the added benefit of reduced program coding, size, and maintenance.

Designate the matrix of distances between n nodes by

$$D = (d_{i,j}) \; , \; (i,j = 1,2,.....,n)$$

In particular the matrix of distances between directly-linked nodes is designated

$$D^0 = (d^0_{i,j})$$

where: $d^0_{i,j} \rightarrow \infty$ if nodes i and j are not directly-linked

and $d^0_{i,j} < \infty$ if nodes i and j are directly-linked.

The policy matrix is denoted by

$$P = (p_{i,j}) \; , \; (i,j = 1,2,.....,n)$$

and, in particular, the initial policy matrix is

$$p^0 = (p^0_{i,j})$$

where: $p^0_{i,j} = 0$ if i and j are not directly-linked

and $p^0_{i,j} = j$ if i and j are directly-linked.

In Floyd's algorithm each node $k = 1,2,...,n$ is utilized in turn as an intermediate node in the path between all other nodes $i,j = 1,2,...n$. This means that the trial distance using node k is computed using

$$d^{i+k}_{i,j} = d^{k}_{i,k} + d^{k}_{k,j} .$$

If the trial distance is less than the previous distance $d^{k+1}_{i,j}$, then the trial distance replaces $d^{k}_{i,j}$ in the distance matrix D, i.e.,

$$d^{k}_{i,k} + d^{k}_{k,j} \rightarrow d^{k}_{i,j}$$

and, also

$$p_{i,k} \rightarrow p_{i,j}$$

after n interations (k = 0,1,2,...,n-1) on the matrix D, the final distance matrix of shortest paths,

$$D^{n} = (d^{n}_{i,j}) ,$$

and the final policy matrix,

$$P^{n} = (p^{n}_{i,j}) ,$$

are obtained.

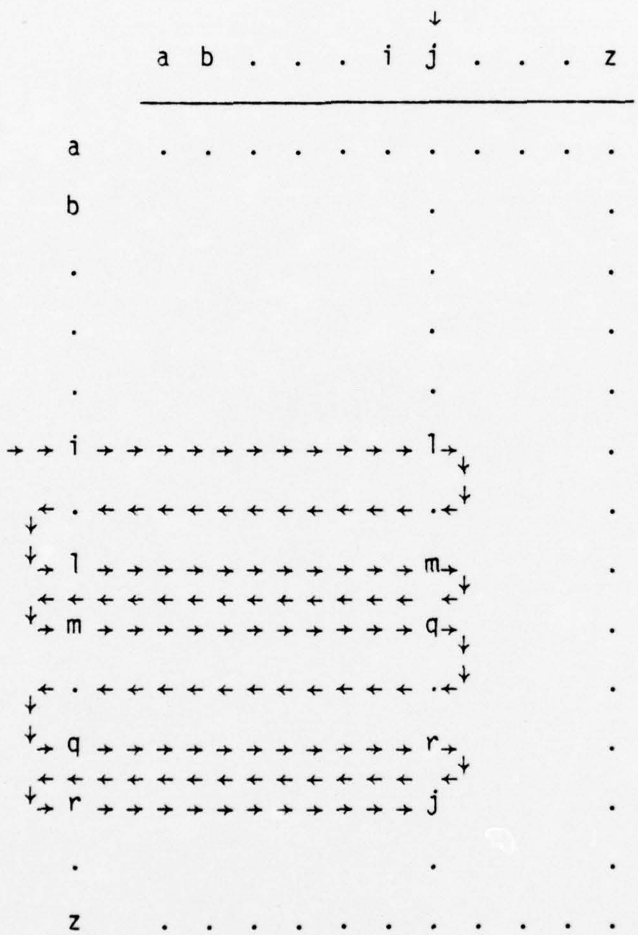
Note that in the computation of the trial distances,

$$d^{k+1}_{i,j} = d^{k}_{i,j} + d^{k}_{k,j} ,$$

and the comparisons to $d^{k}_{i,j}$, only elements of two rows, the ith and the kth, are involved. It is this feature which has been programmed to permit the direct solution of the shortest paths between all unit areas.

The interpretation of a solution policy matrix, P, (dropping the superscript notation) is as follows. For illustration, it is true that the shortest path between two nodes, i and j, involves the following intermediate nodes in order,

i.e., the policy to go from i to m is i to l to m to q to r to j, and also assumes that q is a unit area that will subsequently be damaged. An abbreviated listing of the policy matrix resulting from application of Floyd's algorithm would be as follows:



The partial listing above of a Policy Matrix is used to illustrate a hypothetical policy for the shortest path from node i to node j; i to l, l to m, m to q, q to r, and r to j.

Examination of P to obtain the policy, $p_{i,j}$, would yield in succession

$$p_{i,j} = l \ ,$$

$$p_{l,j} = m \ ,$$

$$p_{m,j} = q \ ,$$

$$p_{q,j} = r \ ,$$

and, finally, $$p_{r,j} = j \ .$$

Therefore, the shortest path policy from node i to node j, as found by examination of p, is given by

$$\bar{p}_{i,j} = (i,l,m,q,r,j) \ .$$

This concludes the interpretation of the policy matrix, $P = (p_{i,j})$.

The following discussions of the damage assessment problem will describe the computational steps required to determine the occurrence of node q as an intermediate node; Dijkstra's procedure for determining the shortest paths between a specified origin and a specified destination; an evaluation of Dijkstra's procedure versus Floyd's procedure; and a feature of Floyd's algorithm which could be implemented with possible benefit.

D. The Damage Assessment Problem

Assume that node q has sustained damage. From the illustrative example,

$$\bar{p}_{i,j} = (i,l,m,q,r,j) \ ,$$

it is true that in the solution policy matrix, P, that

$$p_{m,r} = q$$

$$\text{and} \quad p_{m,j} = q \ .$$

That is, in application, upon reading the mth rows of the policy matrix the value of q will be found to occur two times indicating that q is an intermediate node in the shortest paths, m to r and m to j.

It is preferable to solve only the shortest path, m to r, since the policy

$$\overline{p}_{m,r} = (m,q,r)$$

$$\text{occurs in} \quad \overline{p}_{m,j} = (m,q,r,j) \ .$$

This information is not available at this point; only $p_{m,r} = q$ and $p_{m,j} = q$ are known. To obtain the full policies, m to r and m to j, it is necessary to examine the rth and the jth columns of p which is stored by rows. To examine p by columns (or, equivalently, transpose rows and columns), m dual I/O operations (read and rewind are performed on the complete n x n matrix P) are necessary. Assume this is done and both policies, $\overline{p}(m,r)$ and $\overline{p}(m,j)$ are obtained. Recalling that policies are stated in terms of directly-linked nodes, examination of the qth row of the initial policy matrix, p^0, for those nodes directly-linked to q could show that not only

$$p^0_{q,r} = r \ ,$$

$$\text{but also} \quad p^0_{q,j} = j \ .$$

That is, q is directly-linked to both r and j and both shortest paths must be solved. If instead

$$p^{0}_{q,j} = 0$$

were to occur, it would then be clear to solve only for the shortest path m to r. Thus, transposing the solution policy matrix and reading the initial policy matrix are necessary to determine whether a damaged node is an intermediate node and to determine which shortest paths should be recomputed.

E. Application of Dijkstra's Procedure

In the discussion of the application of Dijkstra's algorithm for the solution of the hypothetical problem, shortest path from node m to node r, the notation will be changed to solve for the shortest path from node 1 to node n. This is done to emphasize that, as in Floyd's algorithm, all 1,2,....,n nodes are used and to facilitate the description.

In addition to the two initial matrices, D^{0} and P^{0}, as previously defined in the description of Floyd's procedure, Dijkstra's procedure uses two vectors;

$\{k_i\}$ = indices of nodes removed from computation of trial distances

and

$\{L_i\}$ = labels associated with each node i = 1,2,....,n.

Initially

$$d_{i,i} = 0 ,$$

$$d_{i,j} = \infty \text{ if nodes i and j are not directly-linked,}$$

$$0 < d_{i,j} < \infty \text{ if nodes i and j are directly-linked,}$$

$$i = 1 ,$$

$$\{k_i\} = \{1\} ,$$

$$\text{and } \{L_j\} = \begin{cases} 0, & j = 1 \\ \infty, & j = 2,3,\ldots\ldots,n . \end{cases}$$

The algorithm proceeds by setting i = 1 and comparing the labels $\{L_i\}$ with the sum of the kth label and direct distance from node k_i to node j:

$$\min (L_j, L_{k_i} + d_{k_{i,j}}) \rightarrow L_j ,$$

for all j not in $\{k_i\}$. Examining the resulting vector of labels $\{L_j\}$ for min $\{L_j\}$ the associated node index, k_{i+1}, is added to the array $\{k_i\}$. This step removes that node from further consideration and places it in the shortest path from node 1 to node n. Then, i is incremented

$$i + 1 \rightarrow$$

and the step

$$\min (L_j, L_{k_i} + d_{k_{i,j}}) \rightarrow L_j , \begin{cases} j = 1,2,\ldots\ldots,n, \\ j \neq \{k_i\} \end{cases}$$

is repeated; min $\{L_j\}$ is found; the associated index added to $\{k_i\}$; and so on until, after, at most, n-1 such iterations, min $\{L_j\}$ is found to be the node, n.

F. Evaluation of Dijkestra's Versus Floyd's Algorithm

In each of the i = 1,2,....,n, m $\leq$ n-1 iterations the direct distance from node k_i to all other nodes j = 1,2,....,n ($j = \{k_i\}$) is needed. As a result, the distance matrix, $(d_{i,j})$, which is stored as a sequential file of rows on disk or tape, may have to be accessed (rewound and/or partially read to row k_i) as many as n times for the hypothetical example of a single damaged intermediate node. These accessings of the file would be necessary for each unique occurrence of a node as a damaged intermediate node and for all such nodes. The un-predictable number of I/O operations plus the number required to transpose the policy matrix are the factors which, in addition to considerations of computer programming simplicity and maintenance, have lead to the decision of preferring Dijkstra's

procedure in favor of Floyd's.

G. Possible Benefit from Implementation of Recomputation Procedure

In Floyd's algorithm, n successive distance matrices, D^{k+1} (k=0,1,1,....,n-1) are computed. The k+1 matrix represents the shortest paths between all node pairs where each node i=1,2,...,k-1,k has been used as an intermediate node. If node q is a damaged intermediate node, the shortest paths may be a recomputed beginning with the matrix in which node q-1 was evaluated as an intermediate node. In other words, all shortest paths based on non-damaged intermediate nodes 1,2,...,q-1 are still valid; the algorithm does not need to be repeated for node indices < q.

The technique has not been implemented, but in principle at least, the successive matrices are saved by writing them as a sequential file on disk or tape storage. From a list of the identification numbers of unit areas that have sustained a change in BOS since the last previous time step in the scenario, the minimum (numerical value) is determined, e.g., the value of the minumum is q. The file of saved distance matrices is advanced to the beginning of the matrix corresponding to q-1. That distance matrix, then, is read as the initial distance matrix to begin the shortest paths algorithm with q as the first tiral intermediate node.

There are several reasons for not implementing the technique at this time. One question pertains to the trade-off between increased computer I/O time and simply recomputing the shortest paths beginning at node one. If implemented, the technique would require that the policy matrix be computed and saved on disk or tape in addition to the distance matrix at each iteration, and examined if it is desired to begin the algorithm at the distance matrix corresponding to the minimum damaged intermediate node. As an alternative, the question of the node being an intermediate node could be ignored; simply re-apply the algorithm beginning with the matrix that corresponds to the minimum damaged node. This would obviate the

necessity of computing/saving the n policy matrices and transposing/examining the final policy matrix.

Another question originates from the definition of distances between directly-linked unit areas. If areas A and B are linked and A is damaged, the distance from A to the boundary between A and B is changed while the distance from the boundary to B is not. These two distances are stored in two different files and are added to obtain the distance from A to B. The unresolved questions are at what stage of the iteration and with what additional complexity and increased I/O time are the distance changes and additions to be performed and with what benefit when there are large numbers of damaged areas.

To derive the maximum benefit from the technique, ideally, the area to be damaged (say there is only one for illustration) should be the n-th; only one iteration of the algorithm would be necessary to resolve the shortest paths. If the damaged area were the 1st, then the algorithm must be completely re-iterated, and the technique has been of no benefit. These observations suggest that the ordering of the unit area identification or the structuring of the Links File should have those areas most likely to be damaged and intermediate ordered last. This criterion has impact on and interacts with other aspects of the total overall problem. Since it could not be evaluated it remains an unresolved question.

Finally, data for 8 unit area networks were available for checkout and evaluation. This small amount of data would not permit answering questions of feasibility in the case of much larger sets of unit areas.

IV. DISCUSSION

The discussion of the Local Emergency Operating System (LEMOS) is undertaken in the context of its interactions with ADS, the attack environment, and fire spread models. Any simulation is impossible without them. Initial discussion will focus on the need for a Development, Test and Evaluation (DT and E) plan and, then, suggest perceived needs for both ADS and LEMOS.

If LEMOS is to have a practical role in local CD planning, a plan by which this role can be attained is essential at this time. It may be fairly stated that the Research phase of RDT and E is nearing completion. The Development, Test, and Evaluation phases should begin immediately. During these phases, selected Federal, State, and Local planners should be enlisted to participate in evolving and conducting them. It is absolutely imperative that they have an important impact on the final configuration of the simulation system before it is used operationally to improve the local CD planning function.

While the mainframe of ADS/LEMOS is nearly complete, several members are conspicuously incomplete. First, ADS is without a special resources submodel and an adequate fire spread model. Second, LEMOS is without a communications module to approximate realistically the communications problems within a damaged area. (The current model assumes no communication problems, i.e., not on the problem file and that all needed communication can be completed as needed.) Third, an upgraded version of the DCPA Emergency Medical Model is needed with its resource requirements incorporated into the LEMOS system. The resulting outputs from the epidemiology model are needed inputs to the overall system. Last, but not least, the GENEC system, as described earlier, does not allow "on line" or "interactive" modifications to the run; however, with the current advances in operating systems, this implementation could be incorporated during the DT and E phases, if plans are evolved to do so.

V. CONCLUSIONS AND RECOMMENDATIONS

DCPA is strongly urged to develop appropriate multi-year program plans to conduct Development, Test, and Evaluation phases of the TELOS system with participation by Federal, State, and Local planning personnel. Special emphasis should be placed during DT and E phases on the use of Case Study Areas. These plans should include the addition of the following elements during the Development phase:

Local plans Pre-processor,

Interactive Control Procedures,

Fire Spread Model (developed by not interfaced with ADS),

Special Resource Damage Submodel,

Communications Submodel,

Utilities Network Submodel,

Medical/Epidemiology Submodel.

Appendix A

USER'S GUIDE

A. USER'S GUIDE

1. Introduction

This section provides a general description of the procedures required to use the Local Emergency Operating System. It is assumed that this usage will occur as a segment of the TELOS model, under the control of GENEC.

2. Use of Master File

The Master Status File, described in detail in Reference 1, is the major link between ADS and LEMOS within TELOS. The Master Status File contains a set of records for all resources except network resources (e.g., highways and power lines) and their damage states in all the unit areas comprising the zone being studied. (Unit areas are defined in reference 1.) These resources include structures, shelter spaces, personnel (both civil defense and general population), and civil defense assets (including teams, equipment sets, and supplies). Thus, the Master Status File is best described as a temporary data base for the zone being studied.

The Master Status File can be used in two different modes by TELOS: a "static" mode and a "dynamic" mode. The former does not involve the use of LEMOS which is indicated by the term "static" (with respect to civil defense countermeasures). The "dynamic" mode is a multi-pass run of the TELOS using civil defense countermeasures (see fig. 1).

The scenario defines an attack. The weapon sizes, times of burst, and locations are specified for the area being studied. The Locate Submodel produces attack environment data for each unit area. The damage assessment submodel (ADS) calculates the effects of the weapons on the resources in the Master Status File. A descriptive printout is then generated. The "dynamic" mode of TELOS incorporates civil defense countermeasure through LEMOS. Civil defense operations determine the damage response functions and the resulting changes are expressed by improved states of resources in the Master Status File. Reports are generated during this cycle to describe these changes and the benefits derived from them.

3. Executive Control

TELOS is controlled by an executive routine written for the CDC-3600 called the Generalized Executive Control (GENEC); it is operable under both Disk and Drum SCOPE.[1/] The control routine assumes that two tapes are available: the first tape contains all the individual programs and the control records for the SORT utility that comprise TELOS, while the second tape contains the parameters that may be needed by the various programs on the first tape (see fig. 2). The use of GENEC is initiated by control card input, which tells GENEC the number of times that the job stream is to be repeated, the number of jobs in the stream, the position of the first job on the program tape and the position of the first set of parameters on the parameters tape. In addition, GENEC will be able to pass five or more words of data from one routine to the next routine executed.

4. Running the Local Emergency Operating System

The first step in using TELOS with LEMOS is the selection of the zone(s) to be studied and the definition of the scenario to be used in the study (see fig. 2).

In step two, since TELOS will be controlled by GENEC, the programs comprising TELOS (including those in LEMOS) should be stored on the program tape, and the controls required by the system should be stored on the parameters tape. The programs in LEMOS require two different types of parameters: "common" parameters and "specific" parameters. The former are used by all the programs in LEMOS, and their values are generally dependent upon the scenario and the purpose of the study. A list of these parameters (or variables) is given in Table I. The "specific" parameters are used to control the functions of each of the programs in LEMOS. There are three "specific" parameters which are input to each program, called RUN-SWITCH, TST-SWITCH, and PRNT-SWITCH. The first of these, RUN-SWITCH, is generally used to select which major functions a program should perform. The second parameter, TST-SWITCH, is generally used for aid in debugging or testing a program. The last parameter, PRNT-SWITCH, is generally used to select printout options within a program, e.g., which file(s) to list.

[1/] A software system operating on the CDC-3600.

A special program, called the CONTROL submodel, is being prepared to run on the CDC-3600. This program will perform two functions: (i) accept the original values of both "common" and "specific" parameters and build the relevant portions of a LEMOS initial parameters tape and (ii) allow changes to the values of the parameters during the execution of the TELOS system. The CONTROL program will accept the values of the parameters from the parameter tape (batch mode) or from the computer console (interactive mode).

In step three, a zone or local area description is prepared in a specific format. The Master Status File, which contains the area's resources except for network data, is prepared on tape. A second tape is prepared containing network data.

In the fourth step, a job card is prepared together with the control card described in 3 above.

Finally, in the fifth step the cards and tapes are submitted for a run on the CDC-3600.

If a run is to be interrupted at selected points in the scenario, the control cards reflect this decision and two or more runs are submitted.

TABLE I: "COMMON" PARAMETERS IN LEMOS

1. Hours from beginning of scenario.
2. Duration of current period, in hours.
3. Sequence number of current period.
4. Flag to indicate whether or not there was a previous period.
5. Code for minimum PF level for shelter spaces to be used.
6. Code for maximum height for shelter spaces to be used.
7. Low radiation level (RADS) for defining the basic operating situation (BOS).
8. High radiation level (RADS) for defining BOS.
9. Low fire level (fraction of area aflame) for defining BOS.
10. High fire level (fraction of area aflame) for defining BOS.
11. Codes for defining five depths of debris.
12. Zone number of area being studied.
13. Level of PF provided by being in automobile.
14. Length of work shift, in hours.
15. Fraction of casualties who are ambulatory, for fifteen (15) injury categories.
16. Maximum number of teams to be assigned to one problem.
17. Identification of sanctuary area for this zone.
18. Priority ranking of operations.
19. Code to indicate whether or not CD can appropriate resources from residences.
20. Code to indicate whether or not population is warned.
21. Weights used to compute measure of effectiveness.

Appendix B

TRANSPORTATION PROGRAM DOCUMENTATION

(9) Final rept.,

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 44U-873	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) LOCAL EMERGENCY OPERATING SYSTEM - LEMOS.		5. TYPE OF REPORT & PERIOD COVERED Final Report July 1976
		6. PERFORMING ORG. REPORT NUMBER (14) RTI-44U-873
7. AUTHOR(s) (10) J. W./Dunn, R. N./Hendry R. O./Lyday		8. CONTRACT OR GRANT NUMBER(s) (15) DAHC20-73-C-0253
9. PERFORMING ORGANIZATION NAME AND ADDRESS Research Triangle Institute Post Office Box 12194 Research Triangle Park, North Carolina 27709		10. PROGRAM ELEMENT PROJECT, TASK AREA & WORK UNIT NUMBERS DCPA Work Unit 4126I
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Civil Preparedness Agency Washington, D.C. 20301		12. REPORT DATE (11) July 1976
		13. NUMBER OF PAGES 100 (12) 120p
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same as 11.		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE NA

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same as 16.

18. SUPPLEMENTARY NOTES

Continuation of Earlier Studies

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Civil Defense
Local Operations
Modeling
Executive Control
Minimum Path Algorithm

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Local Emergency Operating System (LEMOS) is a multi-program model within a system of models which are designed as a part of a computer based system to evaluate local operating systems. The Defense Civil Preparedness Agency Computation Center is a co-developer with RTI in the development of ADS/LEMOS. Effort by RTI during the past year has centered around the development of the control and transportation

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

407926

submodels and at the same time has continued to improve the procedures within other submodels.

This report describes the essential features of the control and transportation models. In addition, a brief discussion is included covering the application of the ADS/LEMOS model to the local Civil Defense planning.

The report concludes that emphasis should shift now from design to development but with the specific objective of gaining support for continued evolution to an operational state.

Adequate developmental testing is recommended before any demonstrations of the planning role are undertaken.

TRANSPORTATION SUBMODEL

Section I: Abstract and Run Description

Abstract

Inputs to the Transportation Submodel comprise the Links File and Control File, both prepared manually by the user, and the Resource File which is generated by the Problem Definition Submodel. The Links File describes the network in each Unit Area in its initial state. The Control File is a list of Unit Areas, possibly a subset of the total, selected for processing. The Resource File indicates changes in state of each Unit Area via the data variable called Basic Operating Status, BOS.

The program computes the shortest paths between all nodes in a Unit Area network and between all Unit Areas.

The primary output is the TVL-REC file, a list of the minimum travel times between all Unit Areas defined in the resource file. Other outputs are: print-outs of the distance and policy matrices before and after the shortest paths are computed; various working or temporary files; and files, either temporary or permanent, of the distance and policy matrices, according to the user's desires and available computer installation options.

Run Description

History

In order to affect civil defense countermeasures, resources, equipment, and personnel must be dispatched to areas requiring remedial action in time to be of benefit. The need for knowing the route of minimum travel time between the origin of the resources and the destination in need of remedial action is self-apparent. Also apparent is the need to choose between competing demands and alternate available resources. The solution of this complex allocation problem requires knowledge of the relative minimum travel times to dispatch resources to demands.

Specifications and Memoranda

The program is written in FORTRAN IV. It has been compiled in FORTRAN-H, link edited, and executed on an IBM 370/175 computer using OS/360, Level 21.6. While every attempt was made to avoid IBM-specific features of the language in order to permit use of the program on other machines with FORTRAN compilers,

some changes may still be necessary especially in regard to I/O statements.

An estimate of available computer core memory at the ultimate user's installation indicated that the program be written so that in execution it would require not more than 250 x 2^{10} bytes (1 byte = 8 bits) on the IBM 370. This specification has been met exclusive of I/O buffers and out-of-core (disk or tape) data storage requirements.

Procedure

See Section 5, Operating Instructions

Method

The program uses Floyd's method for determining the paths of shortest distance between all nodes in a network. See "An Appraisal of Some Shortest-Path Algorithms" by Stuart E. Dreyfus in ORSA, Vol. 17 #3, 1969.

Formulae Used

Given the n x n initial matrix of directly linked nodes in a network:

$$D^0 = (d^0_{i,j})$$

where $d^0_{i,i} = 0$,

$d^0_{i,j} = \infty$ if nodes i and j are not directly linked,

and $d^0_{i,j} < \infty$ if nodes i and j are directly linked,

Floyd's algorithm computes successively for k = 0,1,2,----,n

$$\min\,(d^k_{i,j},\ d^k_{i,k} + d^k_{k,j}) \longrightarrow d^{k+1}_{i,j},\quad i,j = 1,2,----,n.$$

Restrictions

There are restrictions imposed by FORTRAN dimension statements:

75 = max. number of nodes in unit area network

400 = max. number of unit areas.

These limitations may be changed by changing the dimension statements.

Options

Out of the total number of unit area networks represented in the Links File, only those specified in the Control File will be processed.

Device numbers for files read from IN and written on IOUT may be changed in the BLOCK DATA subprogram.

The Control File is read from device IN. Print-out reports are written to device IOUT. Other data files use device numbers set in the program. A complete description is in Section 3, Input-Output Description.

Accuracy

The distances between nodes are rounded to the nearest tenth of a mile. Computed averages of these distances are rounded to the nearest tenth of a mile.

Acknowledgements

Floyd's algorithm is described by Stuart E. Dreyfus in "An Appraisal of Some Shortest Path Algorithms", ORSA, Vol. 17, #3, 1969. Program contributors, either formerly or currently with Research Triangle Institute, include Robert N. Hendry, Russell O. Lyday, Dora B. Wilkerson, Richard J. Coppins Lynn S. Irish, and William J. King. The program author is J. W. Dunn, Research Triangle Institute.

Anticipated Usage

The Transportation Submodel is one within the multi-program model Local Emergency Operating System (LEMOS), which is to be integrated into the Test and Evaluation of Local Operating Systems (TELOS) model, which, in turn, is to function under the control of an executive routine, GENEC.

The Transportation Submodel comprises two major steps. The first prepares a data file of distances between unit areas for use by the second. It is anticipated that the first will be used as a stand-alone program, even possibly outside GENEC. The transportation network data base (Links File) is developed by Unit Areas over a period of time for a geographical study area. Each Unit Area network as it is developed can be processed by the first program and the results analyzed off line for consistency by comparison with the data source documents, topographic or highway maps. Subject to any changes or corrections, the verified results may then be added to the data file input for the second program. The second step computes shortest paths between unit areas. It is anticipated that it will be used in a time-iterative scenario under the control of GENEC.

Timing Factors

Execution time in seconds by CPU and I/O for the case of 8 unit areas containing a total of 120 nodes and from 6 to 26 nodes per unit area was:

	Actual Time in Seconds	
	CPU	I/O
1st Step (find all shortest paths in all unit areas)	10.2	28.5
2nd Step (find all shortest paths between all unit areas)	0.6	9.5

To obtain timing estimates of a larger scale problem, say 400 unit areas, the above times multiplied by 50 (= 400/8) are interpreted subject to the following considerations.

First of all the technique of simple one-point extrapolation is questionable, but it is the only data available.

The I/O times include time required to buffer the print-out of all distance and policy matrices before and after execution of the shortest paths algorithm. Actual printer time is not included. These print-outs would not occur in regular usage.

The CPU and I/O times above were obtained on an IBM 370 operated under MVT (multiprogramming with a variable number of tasks.) The number of other jobs and the job mix in the system has some effect on these times. In a different environment or on a different computer dedicated to the task, different results would occur.

The following time estimates for a 400 unit area problem are felt to be overly pessimistic, except for the second step I/O time which is probably too low.

	Estimated Time in Seconds	
	CPU	I/O
1st Step	510	1,425
2nd Step	30	475

Section II: Flow Charts

Narrative descriptions in lieu of flow diagrams are provided for the following utility type programs.

1. Subroutine RANK

Subroutine RANK is used to rank order the elements of a vector array in ascending order. In particular, it is used in NETWRK to rank the node identification numbers of a given unit area network. The purposes of rank-ordering are to facilitate inspection of the results of data preparation and shortest paths computations and to determine the method of finding equal node i.d.'s in two unit area networks. The vector of unranked values is input to the subroutine; the ranked values are returned in a second vector. A third vector associates the new index of a ranked value with its original index before ranking. Briefly, to illustrate, consider unranked values (c,b,a) returned as (a,b,c):

Original Order	Value	Ranked Order	Value
1	c	3	a
2	b	2	b
3	a	1	c

Thus, the elements of the third array, II, are:

$$II(1) = 3, \; II(2) = 2, \; II(3) = 1.$$

2. Functions COLSUM and ROWSUM

These two functions operate on a given matrix, $(d_{i,j})$, to compute either

$$COLSUM = \sum_{i=1}^{n} d_{i,j}$$

or

$$ROWSUM = \sum_{j=1}^{n} d_{i,j}$$

In particular, these functions are used in ALLNET to compute the total distance from all origins, i, to a specified destination, j, or from a specified origin, i, to all destinations, j, where $(d_{i,j})$ is the matrix of shortest distances between all i and j. In either case the specified node is a boundary node between two unit areas.

3. Function STORE and Function UNMASK.

Function STORE and its entry type function, UNMASK, are used respectively to compress and extract the five hierarchial identifications of a unit area network in a single word as summarized in the following table.

		Bit-Fields					
		Single-word bit configuration					Max Magnitude
Identification	Vector	1	2	3	4	5	
Zone	HIARCH(1)	/////					2^5-1
EOC	HIARCH(2)		//////				2^6-1
Group	HIARCH(3)			//////			2^6-1
Sector	HIARCH(4)				//////		2^6-1
Unit Area	HIARCH(5)					/////////	2^9-1
Total bits		5	6	6	6	9	32
Inclusive Positions		32-28	27-22	21-16	25-10	9-1	

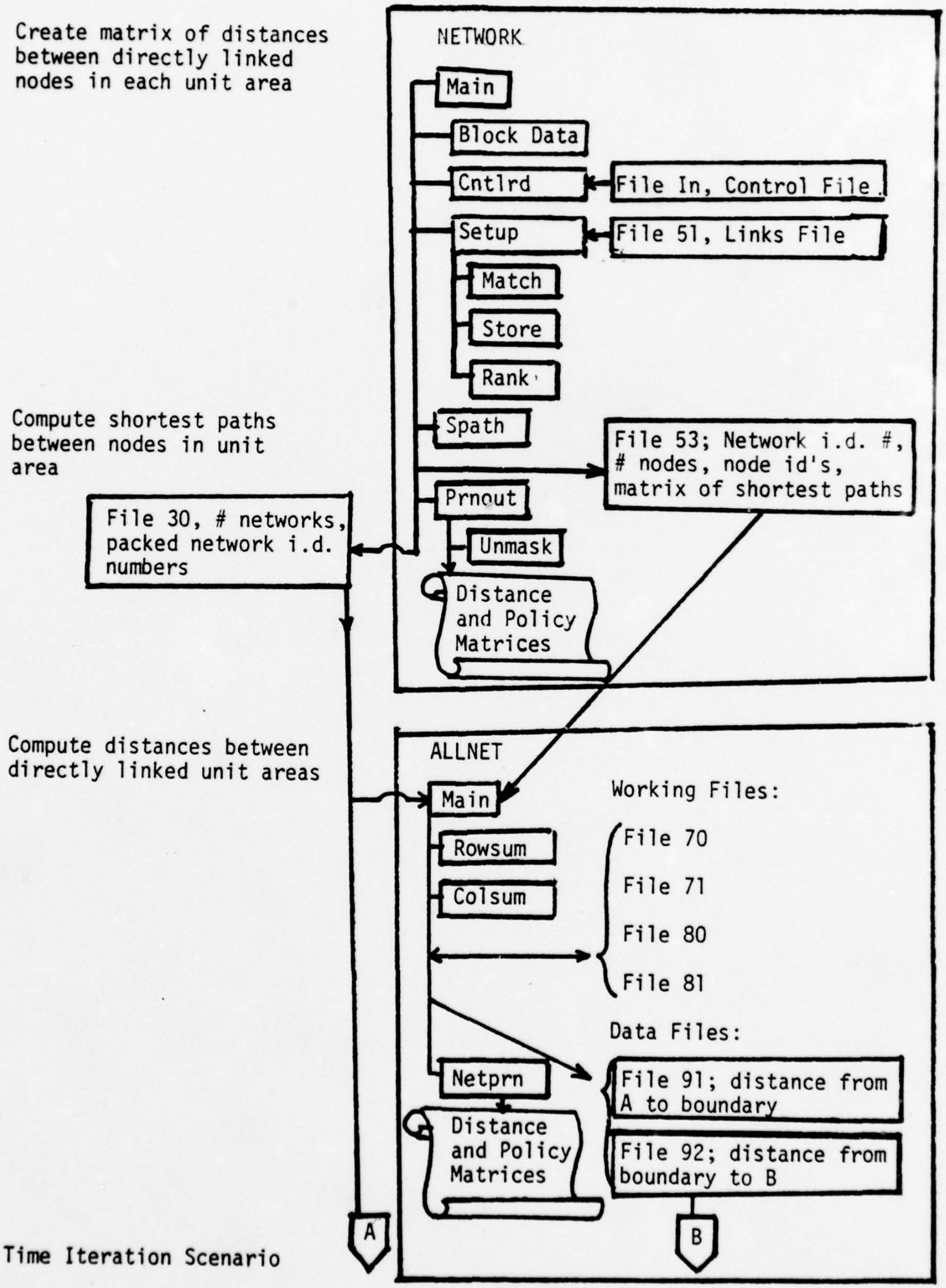
To illustrate, the EOC identification of a unit area stored in the array HIARCH(2), is stored in bit positions 27-22 in the single word NETS (I) for a specified Ith network. This is accomplished by multiplying HIARCH (2) by 2^{21} (which moves the value to the proper bit-field) and then computing the logical sum of the result and NETS (I). To extract the EOC from NETS (I), the logical product of NETS (I) and the 2nd. bit-field configuration above is computed and then the result, in bit positions 27-22, is divided arithmetically by 2^{21} to move it to the lower order bit positions, 1-6.

The masks (bit-field configurations) and powers of 2 are computed by a first call to STORE. For machines with less than 32-bit words, these values must be re-coded. For machines with greater than 32-bit words, re-coding is unnecessary but it may be desirable in order to increase the maximum permissible magnitudes.

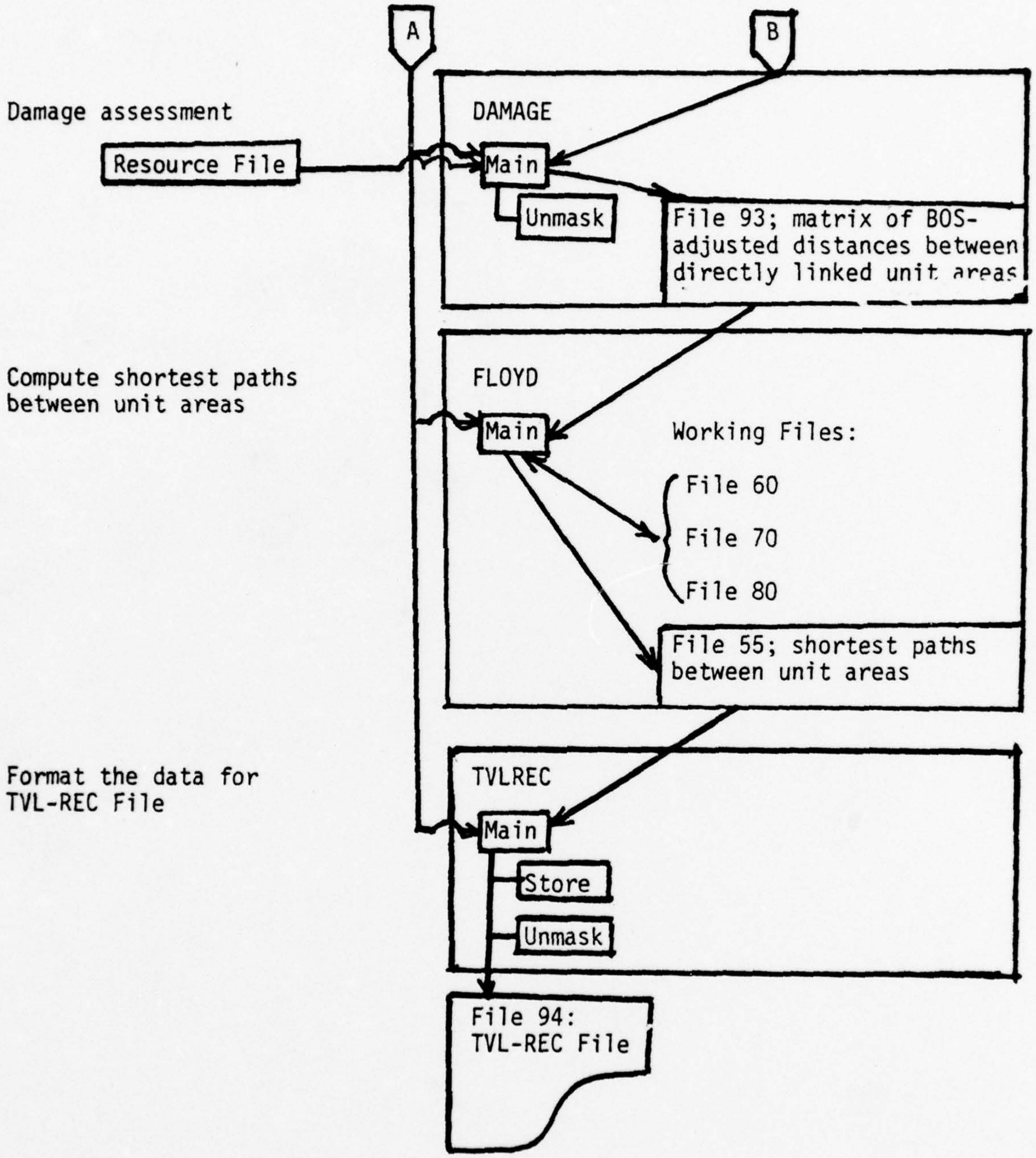
4. Subroutine MATCH

This subroutine compares arithmetically the elements of two integer arrays for equality between any elements. It returns a code of either 0 or 1 indicating either "no" or "yes". If there is equality, the indices of the equal elements are returned. The subroutine is used in subroutine SETUP to create the vector of node identification numbers from the list of nodes contained in the Links File.

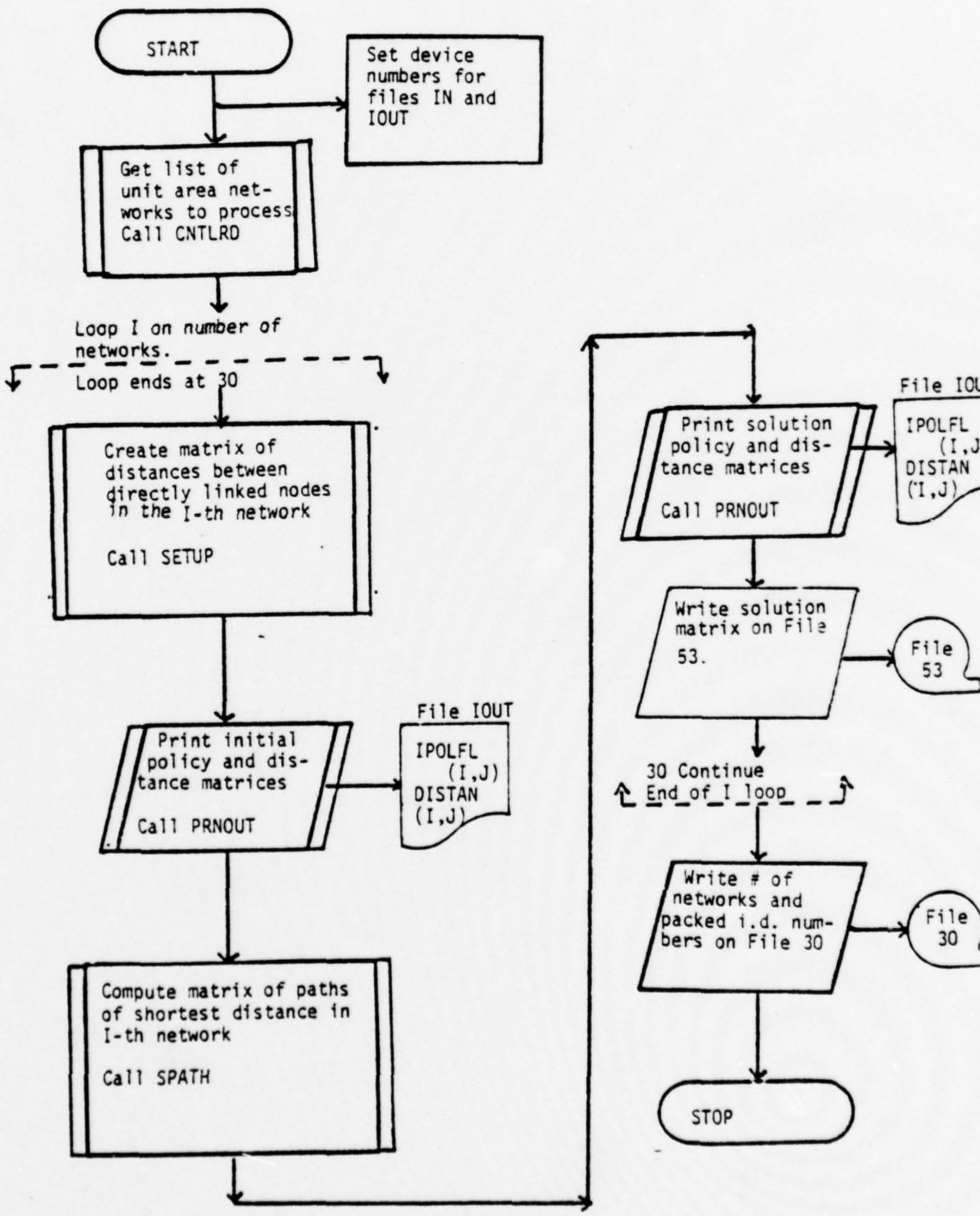
SUBROUTINE AND FILE POSITIONS IN TRANSPORTATION SUBMODEL

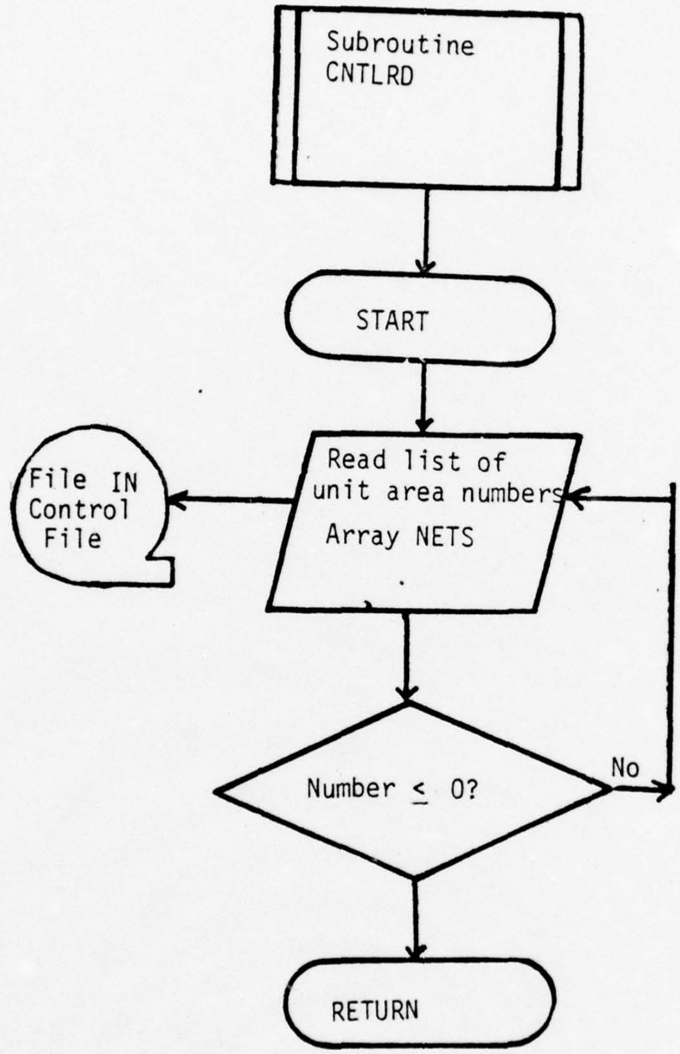


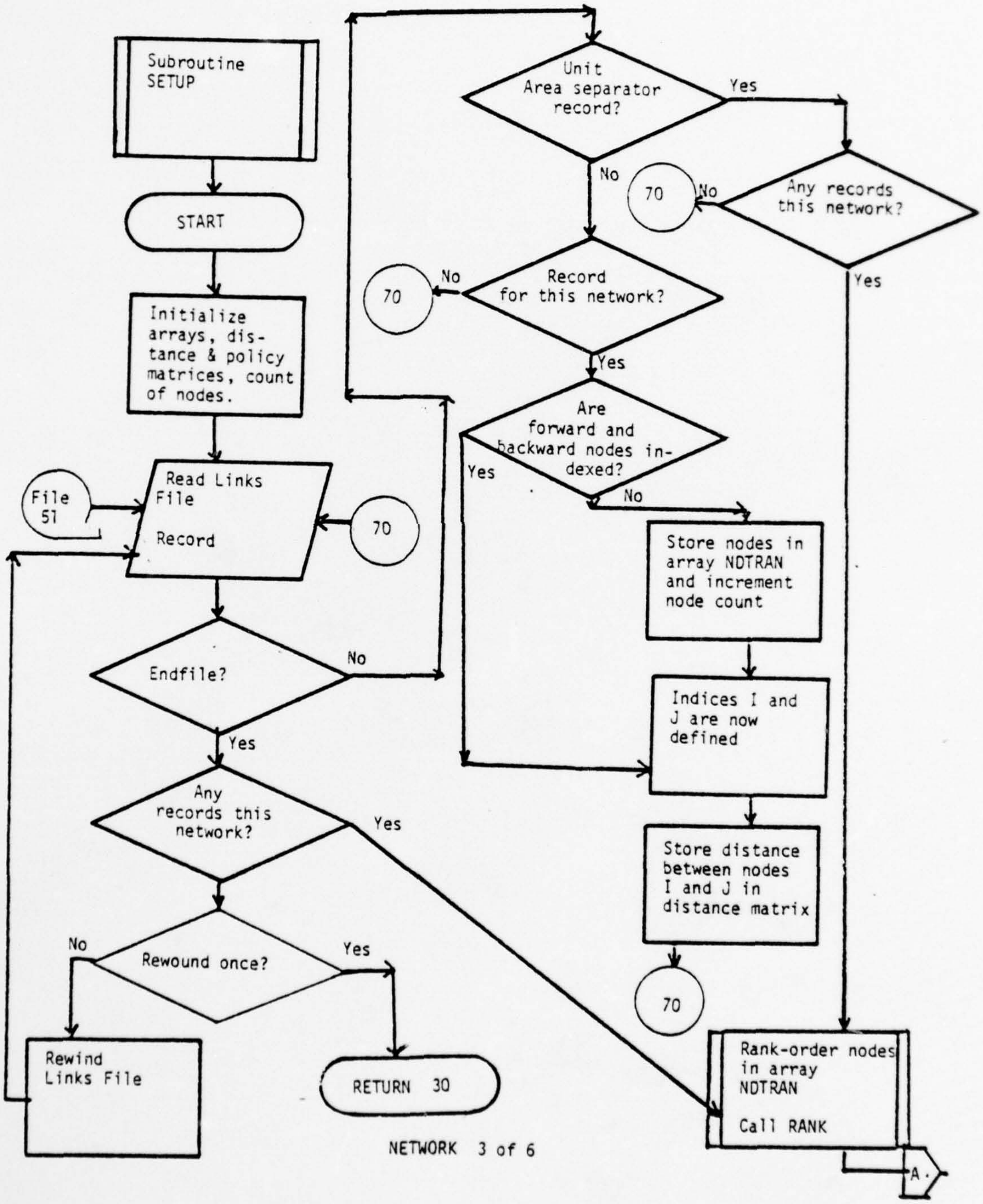
SUBROUTINE AND FILE POSITIONS IN TRANSPORTATION SUBMODEL (Continued)

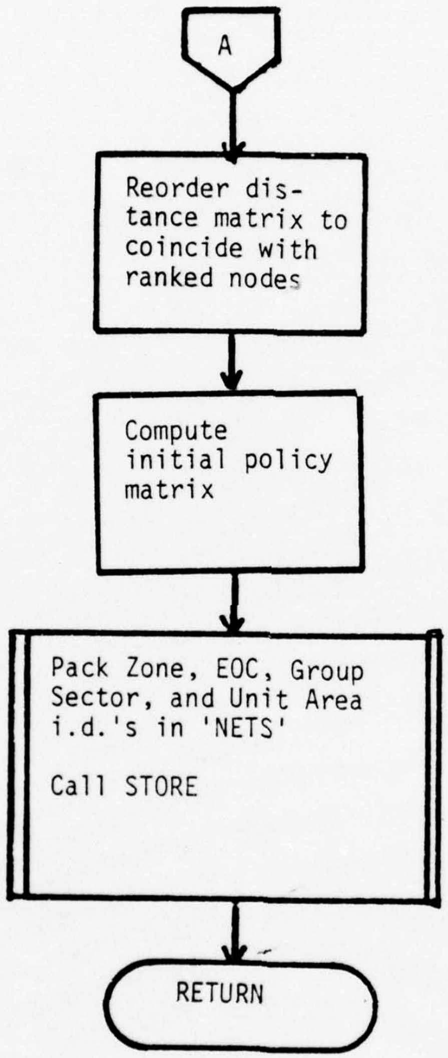


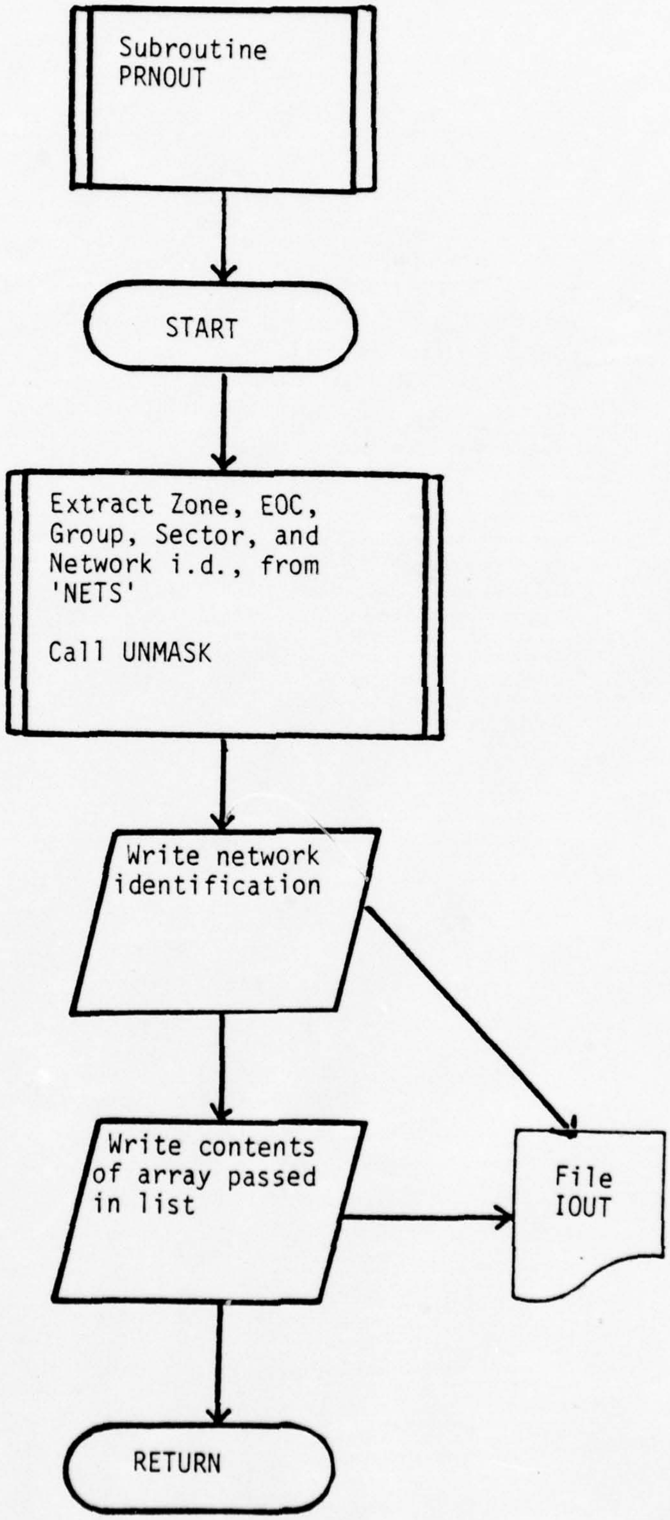
Simplified Flow Chart for NETWORK

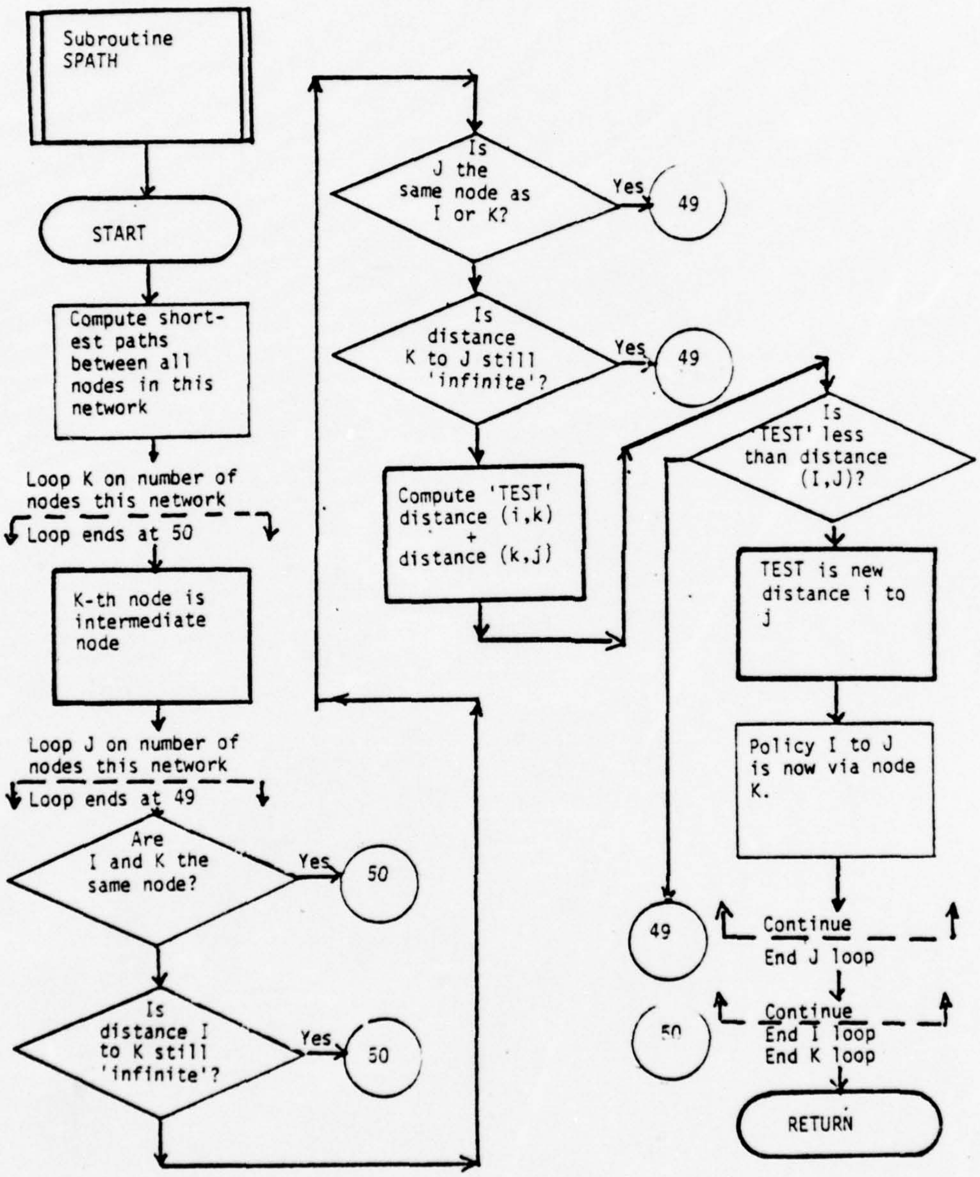



Subroutine
CNTLRD
START
File IN
Control
File
Read list of
unit area numbers
Array NETS
Number ≤ 0?
No
RETURN

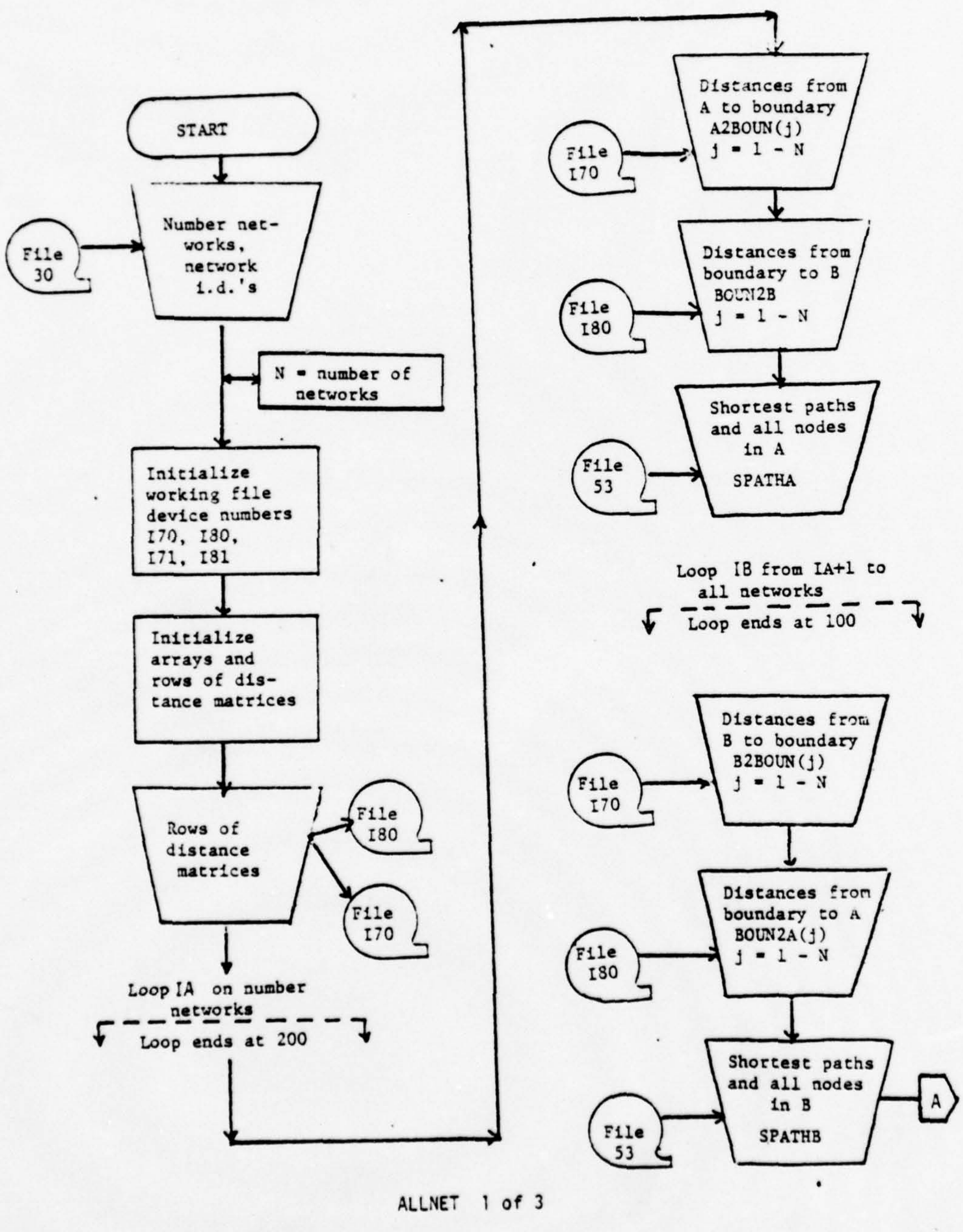

Subroutine SETUP
START
Initialize arrays, distance & policy matrices, count of nodes.
File 51
Read Links File Record
70
Endfile?
No
Yes
Any records this network?
Yes
Rewound once?
No
Yes
Rewind Links File
RETURN 30
Unit Area separator record?
Yes
No
70
No
Any records this network?
Yes
70
No
Record for this network?
Yes
Are forward and backward nodes indexed?
Yes
No
Store nodes in array NDTRAN and increment node count
Indices I and J are now defined
Store distance between nodes I and J in distance matrix
70
Rank-order nodes in array NDTRAN
Call RANK
A

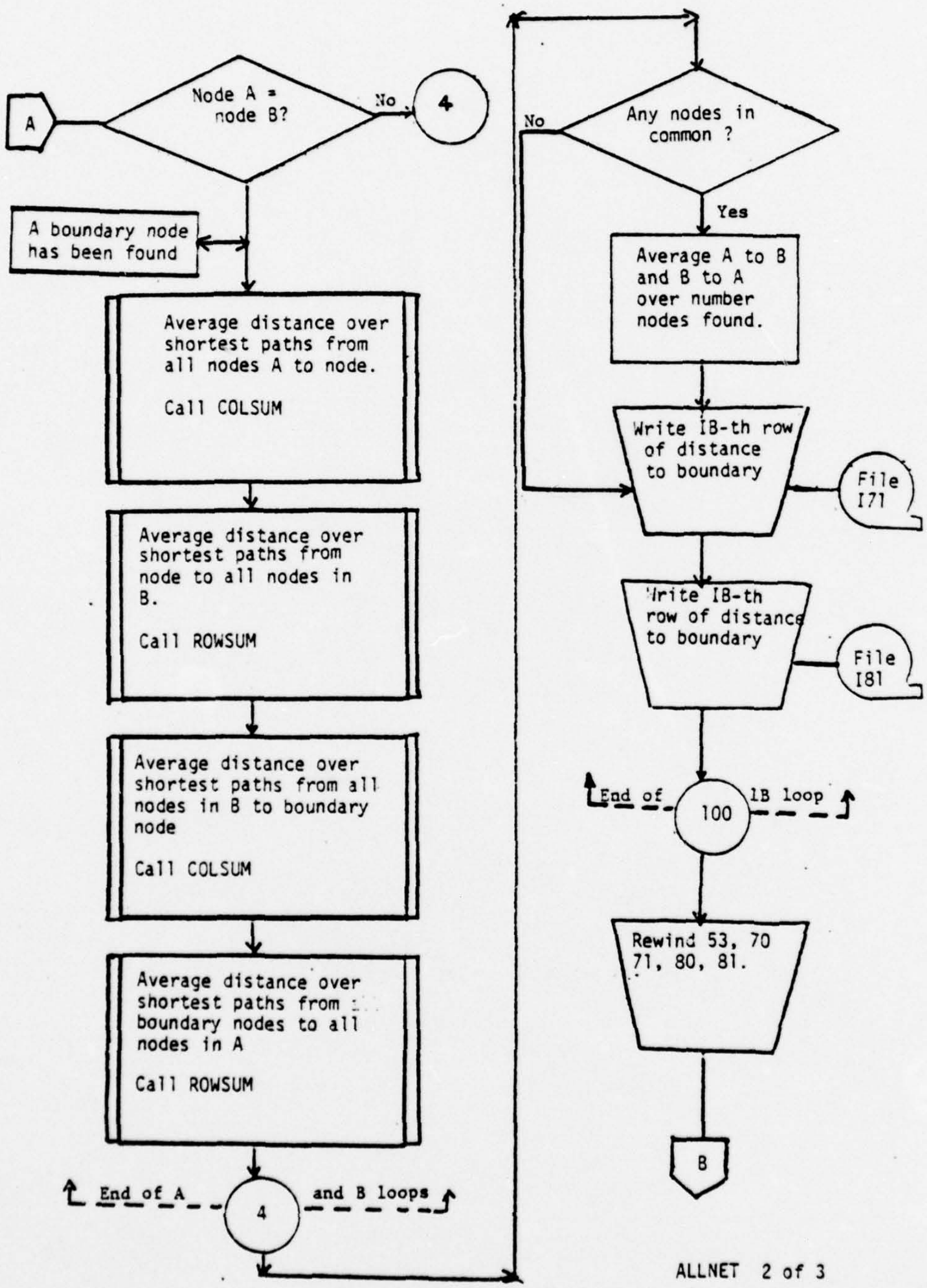

A
Reorder distance matrix to coincide with ranked nodes
Compute initial policy matrix
Pack Zone, EOC, Group Sector, and Unit Area i.d.'s in 'NETS'
Call STORE
RETURN


Subroutine
PRNOUT
START
Extract Zone, EOC, Group, Sector, and Network i.d., from 'NETS'
Call UNMASK
Write network identification
Write contents of array passed in list
File IOUT
RETURN

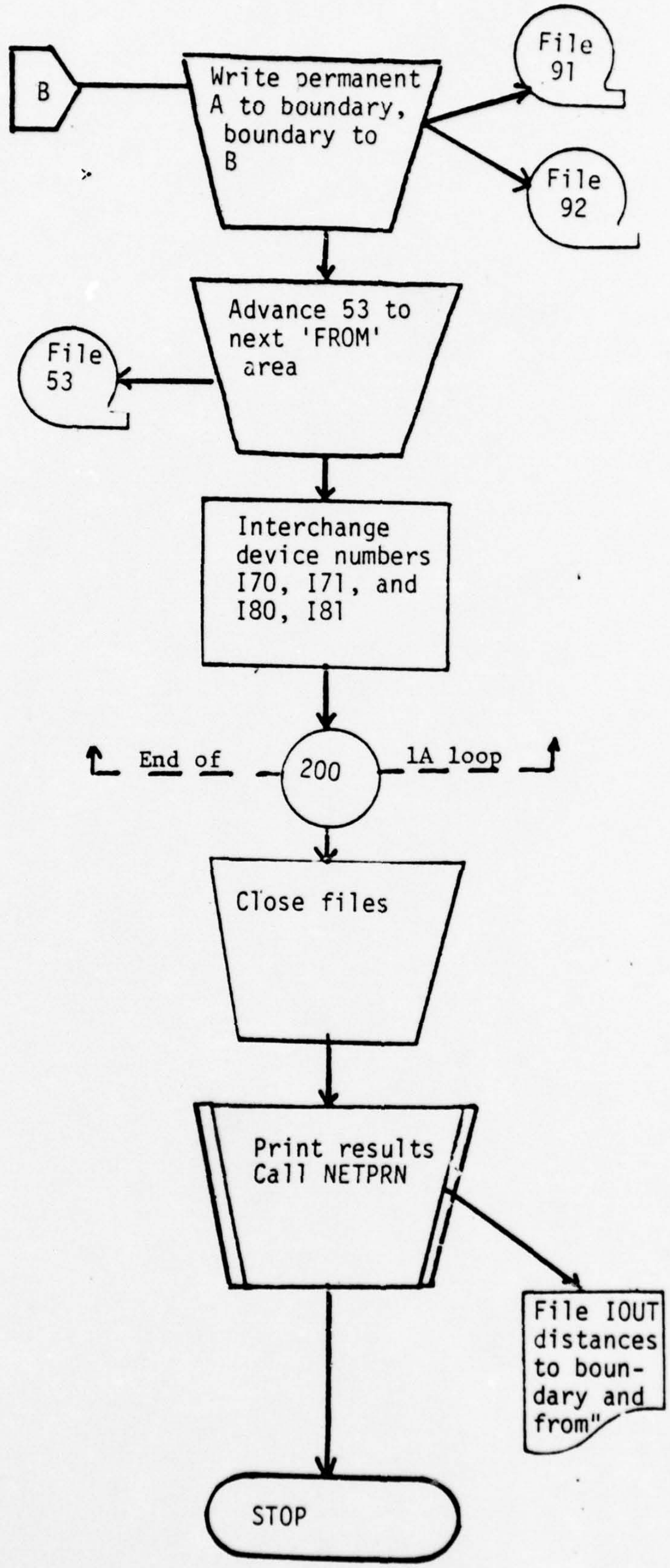

Subroutine SPATH
START
Compute shortest paths between all nodes in this network
Loop K on number of nodes this network
Loop ends at 50
K-th node is intermediate node
Loop J on number of nodes this network
Loop ends at 49
Are I and K the same node?
Yes
50
Is distance I to K still 'infinite'?
Yes
50
Is J the same node as I or K?
Yes
49
Is distance K to J still 'infinite'?
Yes
49
Compute 'TEST' distance (i,k) + distance (k,j)
Is 'TEST' less than distance (I,J)?
TEST is new distance i to j
Policy I to J is now via node K.
49
Continue
End J loop
50
Continue
End I loop
End K loop
RETURN

Simplified Flow Chart for ALLNET



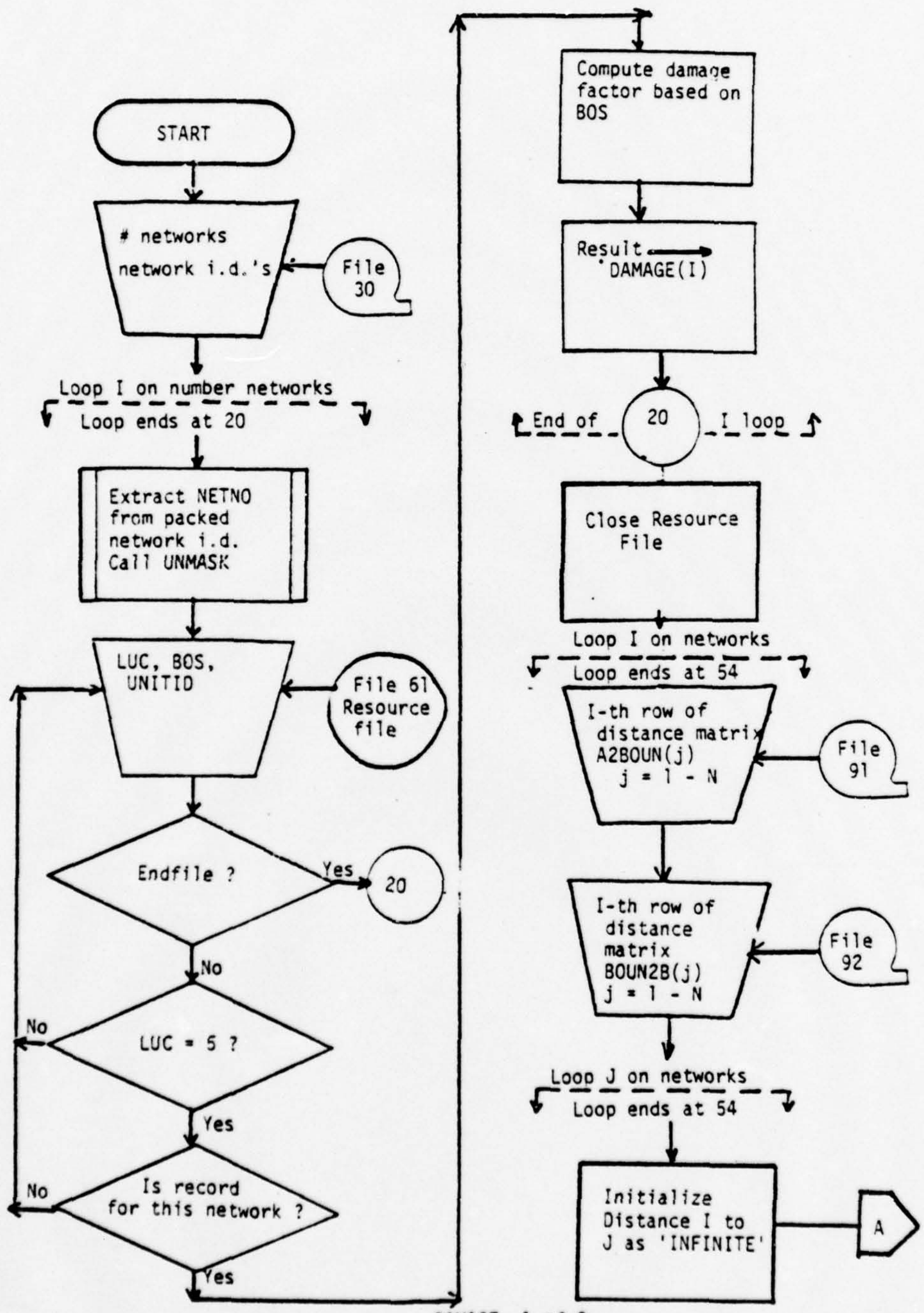

A
Node A = node B?
No
4
A boundary node has been found
Average distance over shortest paths from all nodes A to node.
Call COLSUM
Average distance over shortest paths from node to all nodes in B.
Call ROWSUM
Average distance over shortest paths from all nodes in B to boundary node
Call COLSUM
Average distance over shortest paths from boundary nodes to all nodes in A
Call ROWSUM
End of A
4
and B loops
No
Any nodes in common ?
Yes
Average A to B and B to A over number nodes found.
Write IB-th row of distance to boundary
File I71
Write IB-th row of distance to boundary
File I81
End of
100
IB loop
Rewind 53, 70 71, 80, 81.
B

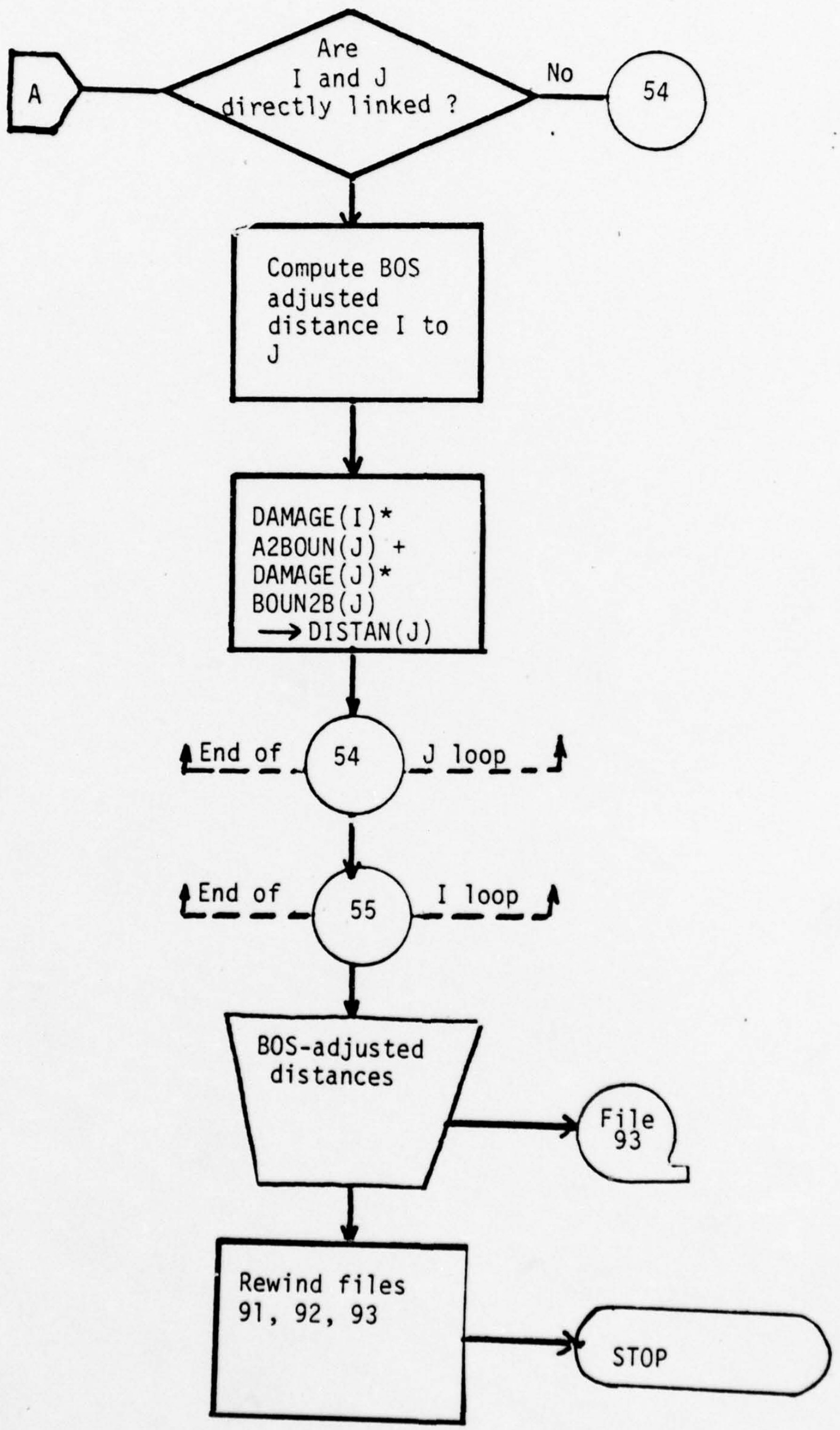
ALLNET 2 of 3


B
Write permanent A to boundary, boundary to B
File 91
File 92
Advance 53 to next 'FROM' area
File 53
Interchange device numbers I70, I71, and I80, I81
End of
200
1A loop
Close files
Print results Call NETPRN
File IOUT distances to boundary and from"
STOP

ALLNET 3 of 3

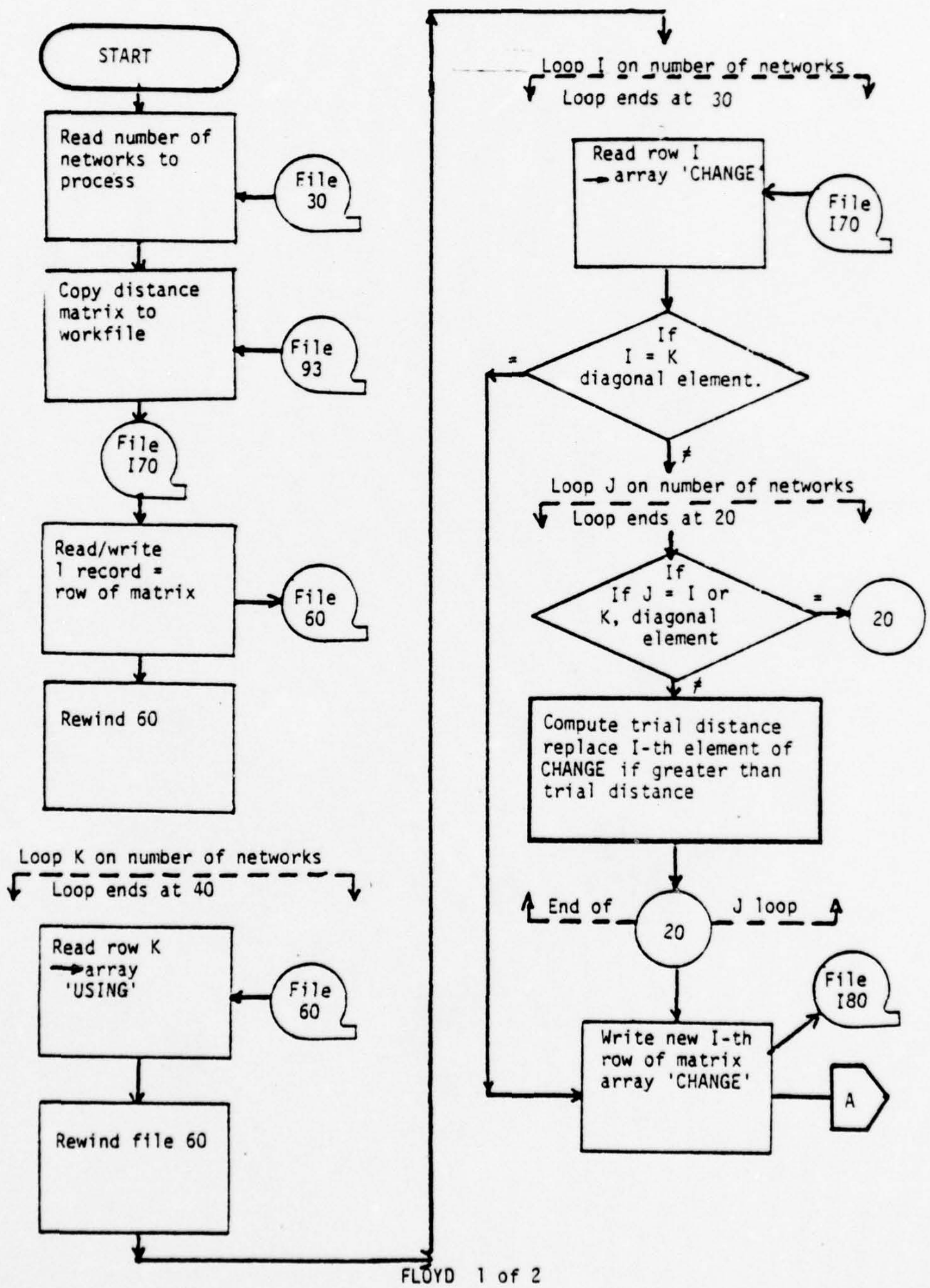
Simplified Flow Chart for DAMAGE

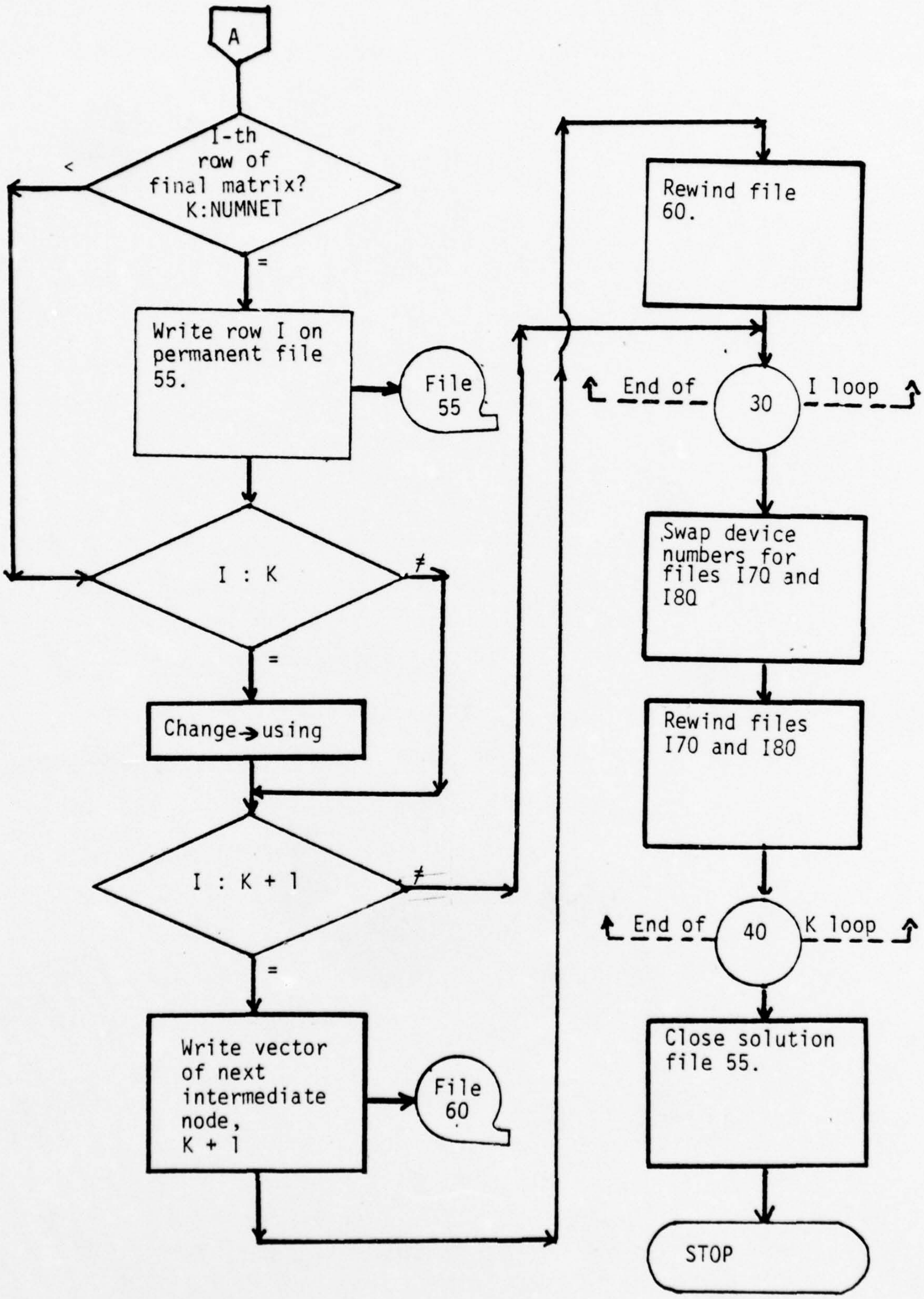



A
Are
I and J
directly linked ?
No
54
Compute BOS
adjusted
distance I to
J
DAMAGE(I)*
A2BOUN(J) +
DAMAGE(J)*
BOUN2B(J)
→ DISTAN(J)
End of
54
J loop
End of
55
I loop
BOS-adjusted
distances
File
93
Rewind files
91, 92, 93
STOP

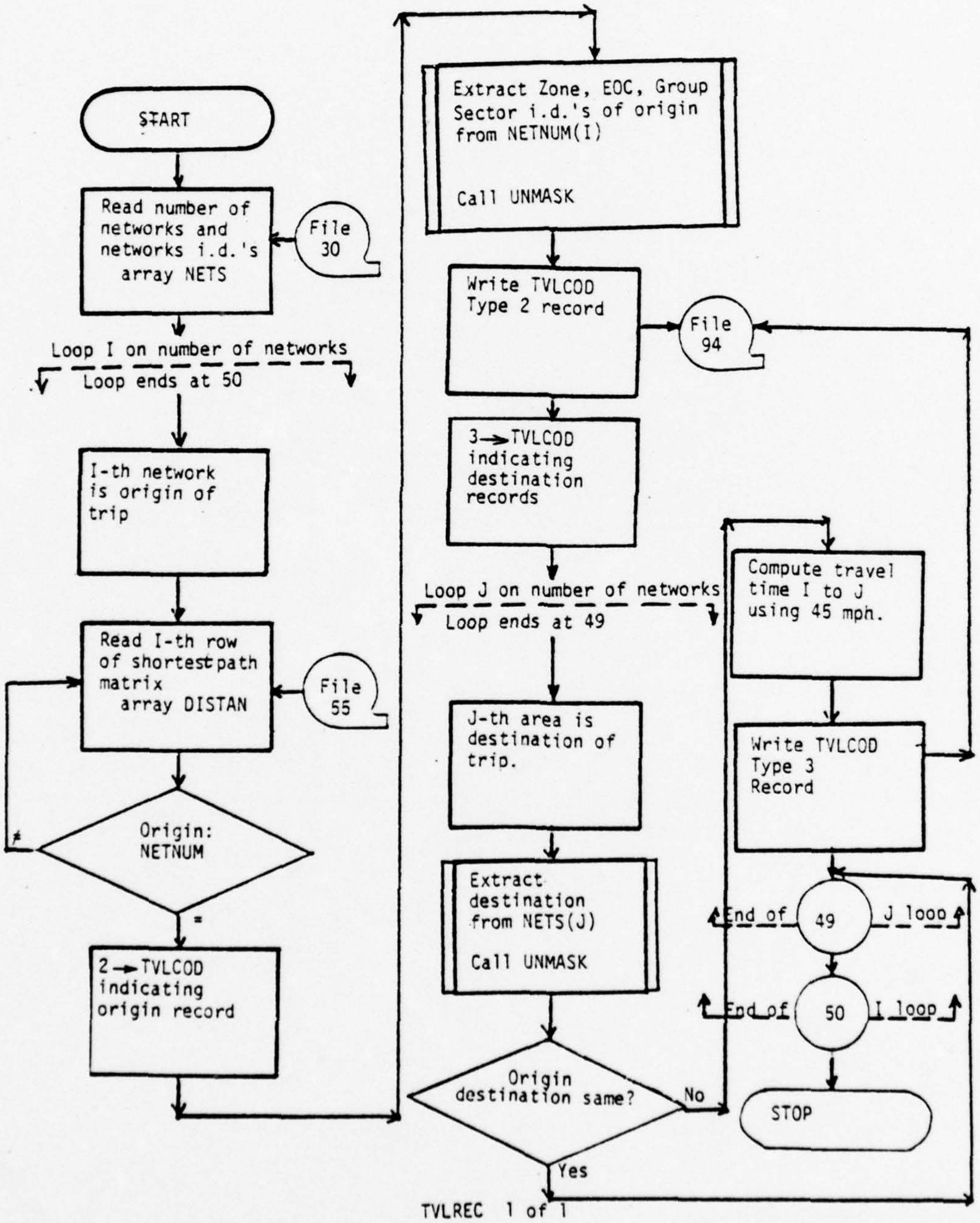
DAMAGE 2 of 2

Simplified Flow Chart for FLOYD




A
I-th row of final matrix? K:NUMNET
<
=
Write row I on permanent file 55.
File 55
I : K
≠
=
Change → using
I : K + 1
≠
=
Write vector of next intermediate node, K + 1
File 60
Rewind file 60.
End of
30
I loop
Swap device numbers for files I70 and I80
Rewind files I70 and I80
End of
40
K loop
Close solution file 55.
STOP

Simplified Flow Chart for TVLREC



Section III: Input-Output Description

The general order of arrangement, creation, and utilization of the 18 files in the Transportation Submodel is shown in Section II, Flow Diagrams. Details concerning the preparation and format of the two manually-prepared files, Control and Links, are contained in the main report. This section will primarily describe the requirements of the submodel-generated files.

All files are sequential files. In the testing and evaluation of the Transportation Submodel, all files except IOUT (the printer) were maintained on on-line disk packs. The REWIND instruction occurs numerous times for closing files not in use to conserve computer core (used for I/O buffers) as well as to reposition files for repetitive processing. The file, IN, which is normally the system input device, i.e. card reader, is also rewound; this could be prohibitive at certain computer installations if the device is not indeed a disk or tape.

Generally, all files are unformatted, exceptions being the files used for prepared input (IN, the Links File, and the Resource File) and formal output (on device IOUT and the TVL-REC File.) Samples of the formatted data files are shown in Section IV, Test Data.

Name	Device	Format	Logical Record Length	Blocksize	No. records as function of no. unit areas, n
Control	IN	FB	80	12960	n/7 1/
Links	51	FB	700	13000	n.a.
	53	VBS	13022	13026	n 2/
IOUT	IOUT	FBA	132	132	n.a.
	30	VBS	1608	11260	1
	70	VBS	1608	11260	n
	71	VBS	1608	11260	n
	80	VBS	1608	11260	n
	81	VBS	1608	11260	n
TOBOUN	91	VBS	1608	11260	n
FRMBOU	92	VBS	1608	11260	n 3/
Resource	61	FB	129	2580	n.a.
	93	VBS	1608	11260	n
	60	VBS	1608	11260	n
	70	VBS	1608	11260	n
	80	VBS	1608	11260	n
	55	VBS	1608	11260	n
TVL-REC	94	FB	20	13020	n^2

1/ 1 record per link for each unit area network + n '999' separator records

2/ Will vary with the type print-out being produced

3/ Depends on number of Land Use Categories in each unit area.

Summary of Data Files Used
in Transportation Submodel

LINKS FILE Record Format[1/]

COBOL Variable	COBOL Format	Fortran Variable(s)	Fortran Format	Card Columns	REMARKS
ZONE	9	HIARCH(1)	I1	1	Zone i.d. number
EOC	9	HIARCH(2)	I1	2	EOC i.d. number
GROUP	99	HIARCH(3)	I2	3-4	Group i.d. number
SECTOR	99	HIARCH(4)	I2	5-6	Sector i.d. number
NETWORK-NO[2/]	999	HIARCH(5)	I3	7-9	Unit Area i.d. number
FWD-NODE	999	IFORW	I3	10-12	Forward node i.d. number
none	x	LVLFOR	I1	13	Level of forward node
BWD-NODE	999	IBACK	I3	14-16	Backward node i.d. number
none	x	LVLBAC	I1	17	Level of backward node
LINK-NO	9999	LINKNO	I4	18-21	Link i.d. number
LINK-NAME	x(11)	not used	23X	22-32	Link name
UA-NO-L	999	not used		33-35	Unit area, left of link
NTWK-NO-L	999	not used		36-38	Network, left of link
UA-NO-R	999	not used		39-41	Unit area, right of link
NTWK-NO-R	999	not used		42-44	Network, right of link
LENGTH	9V9	AMILES	F3.1	45-47	Length of link in miles
WIDTH	9	not used	1X	48	Width of link
TYPE	XX	TYPE	A1	49	Use-type of link
		ITYPE	I1	50	Direction of link
none		not used		50-100[3/]	Room for further expansion; variables for queuing model.

Notes:

1/ This file should be sorted by ZONE, EOC, GROUP, SECTOR, and NETWORK-NO in ascending order.

2/ A record with NETWORK-NO = 999 separates networks.

3/ For development, LRECL = 80 and columns 73-80 contained sequential card identification numbers.

Description of
Links File Records

Link Code	Meaning
S1	Major artery, both ways
S2	Major artery, one-way, forward node to backward node
S3	Major artery, one-way, backward node to forward node
S4	Freeway, both ways
S5	Freeway, one-way, forward node to backward node
S6	Freeway, one-way, backward node to forward node
F1	Firelane
R1	Railroad, single
R2	Railroad, double
R3	Railroad yard
W1	Waterway, small
W2	Waterway, large
W3	Lake

Node Level Code	Meaning
1	Interior to unit area; not shared with other areas
2	Shared between unit areas
3	Shared between sectors
4	Shared Between groups
5	Shared between EOC's
6	On the extreme boundary of zones

Codes Used in
Links File

Control Cards Format

Format	Columns	Variable name
I3	1-3	NETNUM(I); i.d. of unit area to be processed
I1	4	IFLAG - no longer used
I3	5-7	IORIG - no longer used
I3	8-10	IDEST - no longer used

The above format occurs 7 times per card.

A value of NETNUM(I) $\leq$ 0 terminates the string of network numbers

This format is used in Subroutine CNTLRD.

Description of
Control File Records

Test Data

Section IV: Test Data

Test data were derived from maps of the Detroit area for 8 unit areas. Roads of interest were manually transcribed from published maps to a working sheet. Link names and identification numbers for nodes, links, and unit areas were written on the working sheet. These data were transcribed to coding forms in accordance with the Links File format described in the main report with the exception of the inclusion of sequential card numbers in columns 73-80. Listings of the Links File and the Control File follow.

For purposes of development and test, an existing Resource File was modified by truncating the records to 80 columns. To this modified file were added records for LUC = 5 and for other unit areas in the transportation network but not in the Resource File. A listing of the modified Resource File follows.

Printed results before and after the shortest paths computation using the Control, Links, and Resource data files as input follow. Comparison of the distance and policy matrices with the network as drawn on the working sheet is a manual/visual operation. Selecting key links, nodes at extreme points and measuring some distances on the map helped uncover errors in the Links File, aided in de-bugging the program and verified the accuracy of the final results.

There were few predetermined results that could be used except for shortest paths using oodward Avenue or the Chrysler Freeway.

```
40000305012100000000000001200001200001200000100000000000000000000000000000000010
40000307012100000000000001200001200001200000100000000000000000000000000000000020
40000309012100000000000000700000700000700000100000000000000000000000000000000030
40000310012100000000000000100000100000100000100000000000000000000000000000000040
40000311016400344010490029702000700007425000100000100000000000001559302000000050
40000312012100000000000000700000700000700000100000000000000000000000000000000060
40000313012100000000000000400000400000400000100000000000000000000000000000000070
40000317012100000000000000500000500000500000100000000000000000000000000000000080
40000318012100000000000000200000200000200000100000000000000000000000000000000090
40000322011100111000000000400000400000400000100000100000000000000000000000000100
40000324012100000000000000300000300000300000100000000000000000000000000000000110
40000333012100000000000000200000200000200000100000000000000000000000000000000120
40000339012100000000000000200000200000200000100000000000000000000000000000000130
40000340012100000000000000200000200000200000100000000000000000000000000000000140
40000346012100000000000000700000700000700000100000000000000000000000000000000150
40000348018700055017465011199000700002799000100000100000000000000391970000000160
40000349011100006002788007790000700007790000100000100000000000000000000000000170
40000352012100000000000000700000000000000000000000000000000000000000000000000180
40000352022100000000000000000000000000000000000000000000000000000000000000000190
40000353012100000000000000700000700000700000100000000000000000000000000000000200
40000359018700006008750000700000700000000000000001184700000000000000000000000210
40000361011100012000010001200001200001200000100000000000000000000000000000000220
    405   1                                                             00000230
40001405019100000000000000500000500000500000100000000000000000000000000000000240
40001407019100000000000001200001200001200000100000000000000000000000000000000250
40001409019100000000000000700000700000700000100000000000000000000000000000000260
40001410019100000000000000700000700000700000100000000000000000000000000000000270
40001411019100139007836026367000700026367000100000100000000000000000000000000280
40001412019100000000000000700000700000700000100000000000000000000000000000000290
40001413019100000000000000400000400000400000100000000000000000000000000000000300
40001417019100000000000000500000500000500000100000000000000000000000000000000310
40001422019100071000000000400000400000400000100000100000000000000000000000000320
40001424019100000000000000300000300000300000100000000000000000000000000000000330
40001433019100000000000000200000200000200000100000000000000000000000000000000340
40001439019100000000000000200000200000200000100000000000000000000000000000000350
40001440019100000000000000200000200000200000100000000000000000000000000000000360
40001441019100000000000000700000700000700000100000000000000000000000000000000370
40001446019100000000000000700000700000700000100000000000000000000000000000000380
40001448019100040003685012575000700012575000100000100000000000000000000000000390
40001449019100040000466001623000700001623000100000100000000000000000000000000400
40001452019100000000000000700000700000700000100000000000000000000000000000000410
40001453019100000000000000700000700000700000100000000000000000000000000000000420
40001459019100012001000000700000700000700000100007510000000000000000000000000430
40001461019100028000010001200001200001200000100000100000000000000000000000000440
40001463019100000000000000400000400000400000000000000000000000000000000000000450
   2105   1                                                             00000460
   2905   1                                                             00000470
   3105   1                                                             00000480
   7605   1                                                             00000490
   8905   1                                                             00000500
```

4-1 Listing of Records in Input Data File 61, Resource File Detroit Test Case

Page 1 of 1

```
1112 3   3 153 142    7LIVERNOIS    21 21  3  31.0 S1                              00000010
1112 3   3 383 153    7EIGHT MI RD 76 76  3  30.8 S1                              00000020
1112 3   3 783 383    6EIGHT MI RD 76 76  3  30.8 S1                              00000030
1112 3   3 443 783    5EIGHT MI RD 89 89  3  30.8 S1                              00000040
1112 3   3 555 443    4EIGHT MI RD 89 89  3  30.6 S1                              00000050
1112 3   3 624 555    6CONANT AVE  29 29  3  31.1 S1                              00000060
1112 3   3 624 544    4SEVEN MI RD  3  3 29 290.5 S1                              00000070
1112 3   3 544 534    4DEQUINDRE    3  3 29 290.7 S1                              00000080
1112 3   3 534 524    3DEQUINDRE    3  3 14 140.3 S1                              00000090
1112 3   3 524 504    6MCNICHOLS R  3  3 14 140.2 S1                              00000100
1112 3   3 504 414    7MCNICHOLS R  3  3  4  40.4 S1                              00000110
1112 3   3 414 364    8MCNICHOLS R  3  3  4  40.6 S1                              00000120
1112 3   3 364 334    9MCNICHOLS R  3  3  4  40.5 S1                              00000130
1112 3   3 334 834   10MCNICHOLS R  3  3  4  40.3 S1                              00000140
1112 3   3 834 142    1PARKSIDE     3  3 21 211.9 F1                              00000150
1112 3   3 371 142    7SEVEN MI RD  3  3  3  31.3 S1                              00000160
1112 3   3 431 371    6SEVEN MI RD  3  3  3  31.0 S1                              00000170
1112 3   3 544 431    5SEVEN MI RD  3  3  3  30.6 S1                              00000180
1112 3   3 371 383    4WOODWARD AV  3  3  3  31.1 S1                              00000190
1112 3   3 364 371    3WOODWARD AV  3  3  3  31.1 S1                              00000200
1112 3   3 443 431    5CHRYSLER FY  3  3  3  31.0 S4                              00000210
1112 3   3 431 421    4CHRYSLER FY  3  3  3  30.2 S4                              00000220
1112 3   3 504 421    3CHRYSLER FY  3  3  3  30.9 S4                              00000230
      999                                                                         00000240
1123 4   4 414 364    8MCNICHOLS R  3  3  4  40.6 S1                              00000260
1123 4   4 364 334    9MCNICHOLS R  3  3  4  40.5 S1                              00000270
1123 4   4 334 834   10MCNICHOLS R  3  3  4  40.3 S1                              00000280
1123 4   4 292 834    1THOMSON      4  4 31 311.5 S1                              00000290
1123 4   4 285 292    2J.LODGE FWY  4  4 31 310.3 S4                              00000300
1123 4   4 275 285    1J.LODGE FWY  4  4       0.4 S4                              00000310
1123 4   4 315 275    3TUXEDO       4  4       0.2 S1                              00000320
1123 4   4 345 315    2TUXEDO       4  4       0.5 S1                              00000330
1123 4   4 395 345    1TUXEDO       4  4       0.5 S1                              00000340
1123 4   4 321 395    8DAVISON EXP  4  4  4  40.2 S4                              00000350
1123 4   4 351 321    7DAVISON EXP  4  4  4  40.5 S4                              00000360
1123 4   4 401 351    6DAVISON EXP  4  4  4  40.5 S4                              00000370
1123 4   4 863 401    5DAVISON EXP  4  4  4  40.3 S4                              00000380
1123 4   4 315 321    1HAMILTON AV  4  4  4  40.7 S1                              00000390
1123 4   4 321 334    2HAMILTON AV  4  4  4  41.4 S1                              00000400
1123 4   4 345 351    1WOODWARD AV  4  4  4  40.7 S1                              00000410
1123 4   4 351 364    2WOODWARD AV  4  4  4  41.1 S1                              00000420
1123 4   4 395 401    1OAKLAND AVE  4  4  4  40.7 S1                              00000430
1123 4   4 401 414    2OAKLAND AVE  4  4  4  40.9 S1                              00000440
      999                                                                         00000450
112414 14 524 504    6MCNICHOLS R  3  3 14 140.2 S1                              00000460
112414 14 534 524    3DEQUINDRE    3  3 14 140.3 S1                              00000470
112414 14 612 534    1MINNESOTA   29 29 14 140.3 S1                              00000480
112414 14 632 612    1E. NEVADA   29 29 14 140.5 S1                              00000490
112414 14 692 632    2E. NEVADA   29 29 14 141.0 S1                              00000500
112414 14 735 692    3E. NEVADA   29 29 14 141.0 S1                              00000510
```

4-2 Listing of Records in Input Data File 51
Links Data Detroit Test Case

```
112414 14 735 725   1VAN DYKE     14 14       0.5 S1                                  00000520
112414 14 725 815   1MCNICHOLS R  14 14       0.8 S1                                  00000530
112414 14 815 665   1MT ELLIOTT   14 14       0.9 S1                                  00000540
112414 14 665 675   1CANIFF AVE   14 14       0.3 S1                                  00000550
112414 14 675 875   2CANIFF AVE   14 14       0.3 S1                                  00000560
112414 14 875 575   3CANIFF AVE   14 14       0.3 S1                                  00000570
112414 14 575 585   1CONANT AVE   14 14       0.5 S1                                  00000580
112414 14 585 485   1CARPENTER    14 14       0.8 S1                                  00000590
112414 14 485 495   2CARPENTER    14 14       0.3 S1                                  00000600
112414 14 485 511   1CHRYSLER FY  14 14 14 140.5 S4                                  00000610
112414 14 511 863   3DAVISON EXP  14 14 14 140.3 S4                                  00000620
112414 14 511 504   2CHRYSLER FY  14 14 14 140.8 S4                                  00000630
112414 14 601 524   5MCNICHOLS R  14 14 14 141.0 S1                                  00000640
112414 14 821 601   4MCNICHOLS R  14 14 14 140.5 S1                                  00000650
112414 14 681 821   3MCNICHOLS R  14 14 14 140.5 S1                                  00000660
112414 14 815 681   2MCNICHOLS R  14 14 14 140.3 S1                                  00000670
112414 14 591 511   2DAVISON EXP  14 14 14 140.8 S4                                  00000680
112414 14 821 591   1DAVISON E.   14 14 14 140.5 S1                                  00000690
112414 14 601 612   4CONANT AVE   14 14 14 140.3 S1                                  00000700
112414 14 591 601   3CONANT AVE   14 14 14 140.2 S1                                  00000710
112414 14 585 591   2CONANT AVE   14 14 14 140.5 S1                                  00000720
112414 14 632 601   1RYAN STREET  14 14 14 140.5 S1                                  00000730
112414 14 692 681   3MOUND ROAD   14 14 14 140.5 S1                                  00000740
112414 14 681 675   2MOUND ROAD   14 14 14 141.0 S1                                  00000750
     999                                                                              00000760
111221 21 805 305   4FIRELANE           21 211.0 F1                                  00000770
111221 21 795 805   5FIRELANE           21 211.0 F1                                  00000780
111221 21  75 795   9EIGHT MI RD        21 210.5 S1                                  00000790
111221 21 153  75   8EIGHT MI RD  76 76 21 211.0 S1                                  00000800
111221 21 153 142   7LIVERNOIS    21 21  3  31.0 S1                                  00000810
111221 21 834 142   1PARKSIDE      3  3 21 211.9 F1                                  00000820
111221 21 834 134  11MCNICHOLS R  21 21 21 210.9 S1                                  00000830
111221 21 134 124   5LIVERNOIS    21 21 31 310.5 S1                                  00000840
111221 21 124  44   1PURITAN      21 21 31 311.4 S1                                  00000850
111221 21  44 305   6COUZENS FWY  21 21 31 311.1 S4                                  00000860
111221 21  51  44   4WYOMING AVE  21 21 21 210.9 S1                                  00000870
111221 21  61  51   5WYOMING AVE  21 21 21 211.0 S1                                  00000880
111221 21  75  61   6WYOMING AVE  21 21 21 211.0 S1                                  00000890
111221 21 142 134   6LIVERNOIS    21 21 21 211.0 S1                                  00000900
111221 21  61 805   9SEVEN MI RD  21 21 21 210.5 S1                                  00000910
111221 21 142  61   8SEVEN MI RD  21 21 21 210.5 S1                                  00000920
111221 21  51 305  13MCNICHOLS R  21 21 21 211.0 S1                                  00000930
111221 21 134  51  12MCNICHOLS R  21 21 21 211.0 S1                                  00000940
     999                                                                              00000950
112429 29 612 534   1MINNESOTA    29 29 14 140.8 S1                                  00000960
112429 29 632 612   1E. NEVADA    29 29 14 140.5 S1                                  00000970
112429 29 692 632   2E. NEVADA    29 29 14 141.0 S1                                  00000980
112429 29 735 692   3E. NEVADA    29 29 14 141.0 S1                                  00000990
112429 29 745 735   2VAN DYKE     29 29       0.5 S1                                  00001000
112429 29 755 745   3VAN DYKE     29 29       1.0 S1                                  00001010
```

```
112429 29 755 715   1EIGHT MI RD         29 291.0 S1             00001020
112429 29 715 655   2EIGHT MI RD         29 291.0 S1             00001030
112429 29 655 555   3EIGHT MI RD         29 291.0 S1             00001040
112429 29 624 555   6CONANT AVE   29 29   3  31.1 S1             00001050
112429 29 624 544   4SEVEN MI RD   3  3  29 290.5 S1             00001060
112429 29 544 534   4DEQUINDRE     3  3  29 290.7 S1             00001070
112429 29 612 624   5CONANT AVE   29 29  29 290.8 S1             00001080
112429 29 641 624   3SEVEN MI RD  29 29  29 290.5 S1             00001090
112429 29 701 641   2SEVEN MI RD  29 29  29 291.0 S1             00001100
112429 29 745 701   1SEVEN MI RD  29 29  29 291.0 S1             00001110
112429 29 641 632   2RYAN STREET  29 29  29 290.5 S1             00001120
112429 29 655 641   3RYAN STREET  29 29  29 291.0 S1             00001130
112429 29 701 692   4MOUND ROAD   29 29  29 290.5 S1             00001140
112429 29 715 701   5MOUND ROAD   29 29  29 291.0 S1             00001150
      999                                                        00001160
112331 31 855 885   1MEYERS ROAD         31 310.5 S1             00001170
112331 31 845 855   2MEYERS ROAD         31 311.0 S1             00001180
112331 31 845 305   3MEYERS ROAD         31 311.0 S1             00001190
112331 31  44 305   6COUZENS FWY  21 21  31 311.1 S4             00001200
112331 31 124  44   1PURITAN      21 21  31 311.4 S1             00001210
112331 31 134 124   5LIVERNOIS    21 21  31 310.5 S1             00001220
112331 31 111  44   5J.LODGE FWY  31 31  31 311.0 S4             00001230
112331 31 261 111   4J.LODGE FWY  31 31  31 311.1 S4             00001240
112331 31 292 261   3J.LODGE FWY  31 31  31 310.6 S4             00001250
112331 31 285 292   2J.LODGE FWY   4  4  31 310.3 S4             00001260
112331 31 285 245   1GLENDALE     31 31         0.5 S1           00001270
112331 31 245 215   2MONTEREY     31 31         0.7 S1           00001280
112331 31 215 175   1MONTEREY     31 31         0.4 S1           00001290
112331 31 175 185   1DEXTER       31 31         0.5 S1           00001300
112331 31 185  85   1BUENA VISTA  31 31         0.8 S1           00001310
112331 31  85  15   1FULLERTON    31 31         1.0 S1           00001320
112331 31  15 885   2FULLERTON    31 31         0.5 S1           00001330
112331 31  21 855   1SCHOOLCRAFT  31 31  31 310.5 S1             00001340
112331 31  91  21  13DAVISON AVE  31 31  31 311.2 S1             00001350
112331 31 191  91  12DAVISON AVE  31 31  31 310.5 S1             00001360
112331 31 221 191  11DAVISON AVE  31 31  31 310.4 S1             00001370
112331 31 251 221  10DAVISON AVE  31 31  31 310.5 S1             00001380
112331 31 292 251   9DAVISON AVE  31 31  31 310.3 S1             00001390
112331 31  31 845   5FENKELL AVE  31 31  31 310.5 S1             00001400
112331 31 101  31   4FENKELL AVE  31 31  31 311.0 S1             00001410
112331 31 201 101   3FENKELL AVE  31 31  31 310.3 S1             00001420
112331 31 231 201   2FENKELL AVE  31 31  31 310.3 S1             00001430
112331 31 261 231   1FENKELL AVE  31 31  31 310.5 S1             00001440
112331 31  21  15   1WYOMING AVE  31 31  31 310.5 S1             00001450
112331 31  31  21   2WYOMING AVE  31 31  31 311.0 S1             00001460
112331 31  44  31   3WYOMING AVE  31 31  31 310.1 S1             00001470
112331 31  91  85   1LIVERNOIS    31 31  31 310.4 S1             00001480
112331 31 101  91   2LIVERNOIS    31 31  31 311.1 S1             00001490
112331 31 111 101   3LIVERNOIS    31 31  31 310.1 S1             00001500
112331 31 124 111   4LIVERNOIS    31 31  31 310.4 S1             00001510
```

```
112331 31 185 191   2DEXTER        31 31 31 310.2 S1                                00001520
112331 31 191 201   3DEXTER        31 31 31 310.9 S1                                00001530
112331 31 215 221   1LINWOOD       31 31 31 310.6 S1                                00001540
112331 31 221 231   2LINWOOD       31 31 31 310.7 S1                                00001550
112331 31 245 251   112TH          31 31 31 310.6 S1                                00001560
112331 31 251 261   212TH          31 31 31 310.5 S1                                00001570
    999                                                                             00001580
111176 76 383 153   7EIGHT MI RD 76 76  3  30.8 S1                                  00001600
111176 76 783 383   6EIGHT MI RD 76 76  3  30.8 S1                                  00001610
111176 76 783 765   3FIRELANE    89 89 76 762.2 F1                                  00001620
111176 76 765 775   4TEN MILE RD       76 760.9 S1                                  00001630
111176 76 775 165   1FIRELANE          76 760.9 F1                                  00001640
111176 76 165 153   8LIVERNOIS   76 76 76 761.6 S1                                  00001650
111176 76 383 165   5WOODWARD AV 76 76 76 761.7 S1                                  00001660
    999                                                                             00001670
111189 89 443 783   5EIGHT MI RD 89 89  3  30.8 S1                                  00001680
111189 89 555 443   4EIGHT MI RD 89 89  3  30.6 S1                                  00001690
111189 89 565 555   5DEQUINDRE R 89 89      2.0 S1                                  00001700
111189 89 565 475   1TEN MILE RD       89 891.0 S1                                  00001710
111189 89 475 465   2TEN MILE RD       89 890.4 S1                                  00001720
111189 89 465 765   3TEN MILE RD       89 890.1 S1                                  00001730
111189 89 443 451   1CHRYSLER FY 89 89 89 891.1 S4                                  00001740
111189 89 451 465   2STEPHENSON  89 89 89 891.3 S1                                  00001750
111189 89 475 451   1JOHN R. RD  89 89 89 891.2 S1                                  00001760
    999                                                                             00001770
```

```
 30   0   0  40   0   0 140   0   0 210   0   0 290   0   0 310   0   0 760   0   0   00000010
890   0   0  -1                                                                       00000020
```

4-3 Listing of Records in Input Data File IN, Control Card File Identified in JCL as FT01F001. Detroit Test Case, using all unit areas in Links File

Page 1 of 1

Operating Instructions

Section V: Operating Instructions

Since the computer installation at which the Transportation Submodel is to be used is different from the one at which it was developed, only general guidelines are offered on how to assemble and use the FORTRAN source cards. Specific local operating instructions can be developed by the local programming support staff.

The FORTRAN source cards for the Transportation Submodel comprise 5 distinct sets of main programs with associated subroutines. The program names are NETWORK, ALLNET, DAMAGE, FLOYD, AND TVLREC. Three different methods of assembling and using the complete package have been used. One method is to create a separate load module for each of the 5 sets and to execute them in succession as a series of steps in a job. The job control language (JCL) for executing the load modules is shown in Figure 5-1. In creating the load modules provision must be made to provide the multiple-use subroutines, NETPRN, STORE, and UNMASK either as load modules in a job library or by creating duplicate source decks.

In the second manner of assembling the programs, ALLNET, DAMAGE, FLOYD, and TVLREC are compiled as subroutines called by NETWORK. In this case the necessary statements CALL ALLNET, CALL DAMAGE, etc. are added after the statement 30 CONTINUE in NETWORK. The statements SUBROUTINE ALLNET, SUBROUTINE DAMAGE, etc. are added to the source decks and STOP's are replaced by RETURN's. No parameter list is necessary in the calling or subroutine statements. The JCL for a compile-link-edit-go in this situation is shown in Figure 5-2.

The third manner of assembling the programs is a variation on the second described above. NETWORK is converted to a subroutine in the same manner as the other programs. The whole package of subroutines is called by a simply-written main program. The primary feature of the main program is that it contains a loop to simulate time iterations where the Resource File is updated with different values of BOS. This third technique of program assembly is mentioned without an example just to suggest what might be done. In fact, it was done with the programs in an earlier stage of program development where the Master Status File was used instead of the Resource File.

```
//PROCESS JOB RTI.C43.P04956,JWD,M=1,PRTY=1,T=1,FORMS=WHITE
//* RTI.C43.P04956.JWD.PROCESS.CNTL TO EXECUTE TRANSPORTATION SUBMODEL
//* AS A SEQUENCE OF LOAD MODULES IN JOB STEPS.
//S1 EXEC PGM=NETWORK,REGION=98K
//STEPLIB DD DISP=SHR,DSN=RTI.C43.P04956.JWD.NETWORK.LOAD
//FT01F001 DD DISP=SHR,DSN=RTI.C43.P04956.JWD.FT01F001.DATA,
// DCB=BUFNO=1
//FT03F001 DD SYSOUT=A,DCB=BUFNO=1
//FT30F001 DD DSN=RTI.C43.P04956.JWD.NETLIST.DATA,DISP=(NEW,CATLG),
// UNIT=DISK,VOL=SER=RTI222,SPACE=(TRK,(1,1),RLSE),
// DCB=(RECFM=VBS,LRECL=1608,BLKSIZE=1612,BUFNO=1)
//FT53F001 DD DISP=(NEW,CATLG),DSN=RTI.C43.P04956.JWD.PATHS.DATA,
// UNIT=DISK,VOL=SER=RTI222,SPACE=(TRK,(50,50),RLSE),
// DCB=(RECFM=VBS,LRECL=13022,BLKSIZE=13026,BUFNO=1)
//FT51F001 DD DISP=SHR,DSN=RTI.C43.P04956.JWD.LINKS.DATA,DCB=BUFNO=1
//S2 EXEC PGM=ALLNET,REGION=252K
//STEPLIB DD DISP=SHR,DSN=RTI.C43.P04956.JWD.ALLNET.LOAD
//FT03F001 DD SYSOUT=A,DCB=BUFNO=1
//FT30F001 DD DISP=SHR,DSN=RTI.C43.P04956.JWD.NETLIST.DATA,DCB=BUFNO=1
//FT53F001 DD DISP=SHR,DSN=RTI.C43.P04956.JWD.PATHS.DATA,DCB=BUFNO=1
//FT70F001 DD DSN=&&A2BOUN,UNIT=DISK,DISP=(,DELETE),
// SPACE=(TRK,(50,1),RLSE),
// DCB=(RECFM=VBS,LRECL=1608,BLKSIZE=11260)
//FT71F001 DD DSN=&&B2BOUN,UNIT=DISK,DISP=(,DELETE),
// SPACE=(TRK,(50,1),RLSE),DCB=*.FT70F001
//FT80F001 DD DSN=&&BOUN2A,UNIT=DISK,DISP=(,DELETE),
// SPACE=(TRK,(50,1),RLSE),DCB=*.FT70F001
//FT81F001 DD DSN=&&BOUN2B,UNIT=DISK,DISP=(,DELETE),
// SPACE=(TRK,(50,1),RLSE),DCB=*.FT70F001
//FT91F001 DD DISP=(NEW,CATLG),DSN=RTI.C43.P04956.JWD.A2BOUN.DATA,
// SPACE=(TRK,(50,1),RLSE),UNIT=DISK,VOL=SER=RTI222,DCB=*.FT70F001
//FT92F001 DD DISP=(NEW,CATLG),DSN=RTI.C43.P04956.JWD.BOUN2B.DATA,
// SPACE=(TRK,(50,1),RLSE),UNIT=DISK,VOL=SER=RTI222,DCB=*.FT70F001
//S3 EXEC PGM=DAMAGE,REGION=80K
//STEPLIB DD DISP=SHR,DSN=RTI.C43.P04956.JWD.DAMAGE.LOAD
//FT03F001 DD SYSOUT=A,DCB=BUFNO=1
//FT30F001 DD DISP=SHR,DSN=RTI.C43.P04956.JWD.NETLIST.DATA,DCB=BUFNO=1
//FT91F001 DD DSN=RTI.C43.P04956.JWD.A2BOUN.DATA,DISP=SHR,DCB=BUFNO=1
//FT92F001 DD DSN=RTI.C43.P04956.JWD.BOUN2B.DATA,DISP=SHR,DCB=BUFNO=1
//FT93F001 DD DISP=(NEW,CATLG),DSN=RTI.C43.P04956.JWD.DISTANCE.DATA,
// SPACE=(TRK,(50,1),RLSE),UNIT=DISK,VOL=SER=RTI222,DCB=*.FT91F001
//FT61F001 DD DISP=SHR,DSN=RTI.C43.P04956.JWD.RESOURCE.DATA2,
// DCB=BUFNO=1
//S4 EXEC PGM=FLOYD,REGION=100K
//STEPLIB DD DISP=SHR,DSN=RTI.C43.P04956.JWD.FLOYD.LOAD
//FT03F001 DD SYSOUT=A,DCB=BUFNO=1
//FT30F001 DD DISP=SHR,DSN=RTI.C43.P04956.JWD.NETLIST.DATA,DCB=BUFNO=1
//FT93F001 DD DSN=RTI.C43.P04956.JWD.DISTANCE.DATA,DISP=SHR,DCB=BUFNO=1
//FT55F001 DD DISP=(NEW,CATLG),DSN=RTI.C43.P04956.JWD.FINAL.DATA,
// SPACE=(TRK,(50,1),RLSE),UNIT=DISK,VOL=SER=RTI222,DCB=*.FT93F001
```

5-1 Listing of JCL to Execute Transportation Submodel as a Sequence of Load Modules in Job Steps

Page 1 of 2

```
//FT60F001 DD DSN=&&NEXUSE,UNIT=DISK,DISP=(,DELETE),
// SPACE=(TRK,(1,1),RLSE),
// DCB=(RECFM=VBS,LRECL=1608,BLKSIZE=1612)
//FT70F001 DD DSN=&&TEMP1,UNIT=DISK,DISP=(,DELETE),
// SPACE=(TRK,(50,1),RLSE),
// DCB=(RECFM=VBS,LRECL=1608,BLKSIZE=11260)
//FT80F001 DD DSN=&&TEMP2,UNIT=DISK,DISP=(,DELETE),
// SPACE=(TRK,(50,1),RLSE),DCB=*.FT70F001
//S5 EXEC PGM=TVLREC,REGION=100K
//STEPLIB DD DISP=SHR,DSN=RTI.C43.P04956.JWD.TVLREC.LOAD
//FT03F001 DD SYSOUT=A,DCB=BUFNO=1
//FT30F001 DD DSN=&&NETID,DISP=(OLD,DELETE)
//FT30F001 DD DISP=SHR,DSN=RTI.C43.P04956.JWD.NETLIST.DATA,DCB=BUFNO=1
//FT55F001 DD DISP=SHR,DSN=RTI.C43.P04956.JWD.FINAL.DATA,DCB=BUFNO=1
//FT94F001 DD DISP=(NEW,CATLG),DSN=RTI.C43.P04956.JWD.TVLREC.DATA,
// SPACE=(TRK,(50,1),RLSE),UNIT=DISK,VOL=SER=RTI222,
// DCB=(RECFM=FB,LRECL=20,BLKSIZE=13020)
```

5-1 Page 2 of 2

```
//EXECUTE JOB RTI.C43.P04956,JWD,M=1,PRTY=1,T=2
//* RTI.C43.P04956.JWD.EXECUTE.CNTL TO EXECUTE TRANSPORTATION SUBMODEL
//* AS A SEQUENCE OF CALLS WITHIN 'NETWRK' TO THE OTHER PROGRAMS AS
//* SUBROUTINES
//S1 EXEC FTHC
//C.SYSIN DD DSN=RTI.C43.P04956.JWD.NETWORK.FORT,DISP=SHR
//S2 EXEC FTHC
//C.SYSIN DD DSN=RTI.C43.P04956.JWD.ALLNET.FORT,DISP=SHR
//S3 EXEC FTHC
//C.SYSIN DD DSN=RTI.C43.P04956.JWD.DAMAGE.FORT,DISP=SHR
//S4 EXEC FTHC
//C.SYSIN DD DSN=RTI.C43.P04956.JWD.FLOYD.FORT,DISP=SHR
//S5 EXEC FTHCLG,R.G=300K
//C.SYSIN DD DSN=RTI.C43.P04956.JWD.TVLREC.FORT,DISP=SHR
//G.FT01F001 DD DISP=SHR,DSN=RTI.C43.P04956.JWD.FT01F001.DATA,
// DCB=BUFNO=1
//FT03F001 DD SYSOUT=A,DCB=BUFNO=1
//FT30F001 DD DSN=RTI.C43.P04956.JWD.NETLIST.DATA,DISP=(NEW,CATLG),
// UNIT=DISK,VOL=SER=RTI222,SPACE=(TRK,(1,1),RLSE),
// DCB=(RECFM=VBS,LRECL=1608,BLKSIZE=1612,BUFNO=1)
//FT53F001 DD DISP=(NEW,CATLG),DSN=RTI.C43.P04956.JWD.PATHS.DATA,
// UNIT=DISK,VOL=SER=RTI222,SPACE=(TRK,(50,50),RLSE),
// DCB=(RECFM=VBS,LRECL=13022,BLKSIZE=13026,BUFNO=1)
//FT51F001 DD DISP=SHR,DSN=RTI.C43.P04956.JWD.LINKS.DATA,DCB=BUFNO=1
//FT70F001 DD DSN=&&A2BOUN,UNIT=DISK,DISP=(,DELETE),
// SPACE=(TRK,(50,1),RLSE),
// DCB=(RECFM=VBS,LRECL=1608,BLKSIZE=11260)
//FT71F001 DD DSN=&&B2BOUN,UNIT=DISK,DISP=(,DELETE),
// SPACE=(TRK,(50,1),RLSE),DCB=*.FT70F001
//FT80F001 DD DSN=&&BOUN2A,UNIT=DISK,DISP=(,DELETE),
// SPACE=(TRK,(50,1),RLSE),DCB=*.FT70F001
//FT81F001 DD DSN=&&BOUN2B,UNIT=DISK,DISP=(,DELETE),
// SPACE=(TRK,(50,1),RLSE),DCB=*.FT70F001
//FT91F001 DD DISP=(NEW,CATLG),DSN=RTI.C43.P04956.JWD.A2BOUN.DATA,
// SPACE=(TRK,(50,1),RLSE),UNIT=DISK,VOL=SER=RTI222,DCB=*.FT70F001
//FT92F001 DD DISP=(NEW,CATLG),DSN=RTI.C43.P04956.JWD.BOUN2B.DATA,
// SPACE=(TRK,(50,1),RLSE),UNIT=DISK,VOL=SER=RTI222,DCB=*.FT70F001
//FT93F001 DD DISP=(NEW,CATLG),DSN=RTI.C43.P04956.JWD.DISTANCE.DATA,
// SPACE=(TRK,(50,1),RLSE),UNIT=DISK,VOL=SER=RTI222,DCB=*.FT91F001
//FT61F001 DD DISP=SHR,DSN=RTI.C43.P04956.JWD.RESOURCE.DATA2,
// DCB=BUFNO=1
//FT55F001 DD DISP=(NEW,CATLG),DSN=RTI.C43.P04956.JWD.FINAL.DATA,
// SPACE=(TRK,(50,1),RLSE),UNIT=DISK,VOL=SER=RTI222,DCB=*.FT93F001
//FT60F001 DD DSN=&&NEXUSE,UNIT=DISK,DISP=(,DELETE),
// SPACE=(TRK,(1,1),RLSE),
// DCB=(RECFM=VBS,LRECL=1608,BLKSIZE=1612)
// SPACE=(TRK,(50,1),RLSE),DCB=*.FT70F001
//FT94F001 DD DISP=(NEW,CATLG),DSN=RTI.C43.P04956.JWD.TVLREC.DATA,
// SPACE=(TRK,(50,1),RLSE),UNIT=DISK,VOL=SER=RTI222,
// DCB=(RECFM=FB,LRECL=20,BLKSIZE=13020)
```

5-2 Listing of JCL to Execute Transportation Submodel
Using NETWRK as a Driver Routine to Call other Subroutines

Page 1 of 1

Section VI: Suggestions, Warnings, and Changes

Several improvements could be added. At one stage of development the user had to supply the list of nodes connecting unit areas. In an effort to reduce the number of files and the amount of manual data preparation required, it was decided to let the computer do the work of determining unit area connectivity, and this file was eliminated. As experience suggests and familiarity with the program and files increases, it may become desirable to reinstate this list to provide an override option to describe linkage between unit areas.

Another improvement would be to permit changing, say, road type from major freeway both ways to one-way for "movement to shelter" procedures during the course of the time iterative scenario. At present, the Links File which describes the network in its initial state is "outside" the time iterative loop. Thus, the only way to effect a change in states, other than damage, is to so describe the network initially in the Links File.

Another improvement would be to create a second version of the step that computes the shortest paths between unit areas when the total number of unit areas is less than approximately 150. This second version would replace FLOYD, which uses the distance matrix stored on disk or tape, with something similar to SPATH, which stores the entire matrix internally in core, at a great savings in I/O time. Hand calculations suggest that 150 unit areas will generate distance matrices (150 x 150) just within core limitations. This improvement would be justified if there were anticipated a large number of study areas containing less than 150 unit areas. Another advantage to be gained would be that Dijkstra's algorithm could then be added to resolve for origin/destination-specific shortest paths after damage.

Some infrequents used programming techniques are:

(1) the use of 9999 to represent a distance of "infinity" between nodes

(2) storing multiple values in pre-defined bit-fields of a single word (see the description of STORE and UNMASK).

(3) storing the value of distance in the upper half of a word and policy in the lower half for words in the distance matrix. The value 2^{16} is used arithmetically to shift and extract values, and will be seen to occur frequently in the program.

A technique for changing recursively between two device numbers in READ and WRITE statements is used in FLOYD and ALLNET. In order to process data written to an output device, (e.g., number J), the device number in the READ statement must be set to value J, and the new WRITE output device number is set to value I, the prior value of the input device. Thus, if we are reading from I and writing to J, the recursion is:

$$(I + J) - I \longrightarrow I$$
$$(I + J) - J \longrightarrow J$$

The most likely error to occur will be an inacessible node in a unit area network. This condition will be noted by the value 9999 appearing somewhere in the shortest paths matrix. Examine the Links File for that unit area for transposed digits in link and/or node identification numbers, an incorrect node i.d., inclusion of a node from another network, or incorrect specification of a one-way street.

```
C
C  RTI.C43.P04956.JWD.NETWORK.FORT
C  PROGRAM NETWORK IN TRANSPORTATION SUBMODEL.
C
C
C  REMOVE 'C' IN SUBROUTINE NETWRK TO USE AS A SUBROUTINE.  ALSO CHANGE
C  STOP AND RETURN CARDS BELOW.
C
C     SUBROUTINE NETWRK
      COMMON/IO/IN,IOUT
      COMMON/BLOCK1/NINE16,INFIN,KINFIN, ITWO16,NETS(400)
      COMMON/BLOCK2/DISTAN(75,75),IPOLFL(75,75),NOTRAN(75),INDEXT(75)
      INTEGER HIARCH(5), DISTAN
      ITWO16 = 2**16
      NINE16 = 9999
      INFIN = NINE16
      KINFIN = INFIN*ITWO16
C
C  READ IN LIST OF NETWORKS TO BE PROCESSED.
C
      CALL CNTLRD(NUMNET)
C
C  AT THIS POINT, WE HAVE ALL NECESSARY DATA TO BEGIN THE SHORTEST PATH
C      PROCEDURES.  THEY WILL BE IMBEDDED WITHIN A LOOP, SO THAT EACH
C      NETWORK WILL BE ANALYZED IN TURN.
C
      DO 30 I=1,NUMNET
C
C  CREATE MATRIX OF DISTANCES BETWEEN DIRECTLY LINKED NODES IN THE I-TH
C  NETWORK.
C
      CALL SETUP(I,NODES,&30)
C
C  PRINT THE INITIAL POLICY MATRIX BETWEEN NODES FOR I-TH NETWORK ON
C  DEVICE 'IOUT'.
C
      CALL PRNOUT(IPOLFL,NODES,I)
C
C  PRINT THE INITIAL MATRIX OF DISTANCES BETWEEN NODES FOR THE
C  I-TH NETWORK ON DEVICE 'IOUT'
C
      CALL PRNOUT(DISTAN,NODES,I)
C
C  COMPUTE THE PATHS OF MINIMUM DISTANCE BETWEEN ALL NODES IN THE I-TH
C  NETWORK:
C
      CALL SPATH(NODES)
C
C  SAVE THE RESULTS OF THE SHORTEST PATHS COMPUTATION ON FILE 53 FOR
C  SUBSEQUENT PROCESSING OF SHORTEST PATHS BETWEEN UNIT AREAS.
```

7-1 Listing of Fortran Source Program NETWORK

```
C
      WRITE(53)NETS(I),NODES,NOTRAN,DISTAN
C
C  PRINT THE FINAL SOLUTION POLICY AND DISTANCE MATRICES ON DEVICE
C 'IOUT'.
C
      CALL PRNOUT(IPOLFL,NODES,I)
      CALL PRNOUT(DISTAN,NODES,I)
   30 CONTINUE
      REWIND 53
C
C  HAVE PROCESSED NUMNET UNIT AREA NETWORKS.  SAVE THE PACKED NETWORK
C  I.D. NUMBERS, IN NETS(I),I=1,2,...,NUMNET, ON FILE 30.
C
      WRITE(30) NUMNET,NETS
      REWIND 30
C
C  REMOVE 'C' IN COLUMN 1 TO EXECUTE REST OF TRANSPORTATION MODEL
C  PROGRAMS AS SUBROUTINES.
C
C      CALL ALLNET
C     CALL DAMAGE
C     CALL FLOYD
C     CALL TVLREC
      STOP
C
C  TO EXECUTE 'NETWORK' AS A SUBROUTINE REMOVE 'C' IN RETURN CARD,
C  REMOVE 'STOP', AND REMOVE 'C' IN SUBROUTINE NETWORK CARD ABOVE.
C -
C     RETURN
      END
      BLOCK DATA
      COMMON/IO/IN,IOUT
      DATA IN/1/,IOUT/3/
      END
      SUBROUTINE SETUP(IAM,NODES,*)
C***********************************************************************
C  THIS SUBROUTINE READS THE NETWORK DATA FOR NETNUM AND CREATES THE
C  INTIAL MATRIX OF DISTANCES BETWEN DIRECTLY LINKED NODES.
C  IT ALSO PACKS THE ZONE, EOC, GROUP, SECTOR, AND UNIT AREA
C  I.D. NUMBERS INTO NETS(IAM), WHICH ENTERS SETUP CONTAINING ONLY
C  THE NETWORK, E.G. UNIT AREA, I.D. NUMBER.
C  VARIABLES:
C  NOTRAN(I) : ACTUAL NODE NUMBER (THE SHORTEST PATH ROUTINES WILL WORK
C              WITH DUMMY NUMBERS 1, 2, ETC.)
C  DISTAN(I,J): DISTANCE FROM NODE I TO NODE J.
C  IPOLFL     :WORKING STORAGE FOR SORT; BECOMES IPOLFL(I,J) IN 'SPATH',
 C            : THE POLICY MATRIX.
C  INDEXT     : WORKING STORAGE FOR SORT.
C***********************************************************************
```

```
      COMMON/IO/IN,IOUT
      COMMON/BLOCK1/NINE16,INFIN,KINFIN, ITWO16,NETS(400)
      COMMON/BLOCK2/DISTAN(75,75),IPOLFL(75,75),INDEXT(75),NDTRAN(75)
      DIMENSION INODES(75),HIARCH(5)
      INTEGER FOUND,REWIND,ZONEID,EOCID,GRUPID,SECTID,HIARCH,DISTAN
C
C
      IONE = 1
      NETNUM = NETS(IAM)
C
C     INITIALIZE THE DISTANCE MATRIX.
C
      DO 15 I=1,75
      DO 14 J=1,75
      IPOLFL(I,J) = NINE16
   14 DISTAN(I,J)=NINE16
      IPOLFL(I,I) = 0
   15 DISTAN(I,I) = 0
C
C  INITIALIZE THE MATRIX OF NODE I.D.'S
      DO 16 L=1,75
      INODES(L) = L
      INDEXT(L) = 0
   16 NDTRAN(L) = 0
      NODES=1
   65 FOUND=0
      REWIND = 0
C
C     READ THE LINKS FILE - LOOKING FOR NETWORK I.D. NUMBER NETS(IAM)
C
   70 READ(51,80,END=71)HIARCH,IFORW,LVLFOR,IBACK,LVLBAC,LINKNO,AMILES,
     1ITYPE,ITYPE
   80 FORMAT(2I1,2I2,I3,2(I3,I1),I4,23XF3.1,1XA1,I1)
      NETNO = HIARCH(5)
      IF(NETNO.EQ.NETNUM)GO TO 81
      IF(NETNO.EQ.999.AND.FOUND.EQ.1)GO TO 116
      GO TO 70
   71 REWIND 51
      IF(FOUND.EQ.1)GO TO 116
      REWIND = REWIND+1
      IF(REWIND.LT.2)GO TO 70
      WRITE(IOUT,38)NETNUM,NETNUM
      RETURN 1
   38 FORMAT(/' IN SUBR. SETUP.  2ND REWIND ON LINK FILE (DEVICE 51) LOO
     1KING FOR NETWORK NUMBER',I4,'.  NOT FOUND '/' RETURNED TO MAIN WIT
     1HOUT COMPUTING SHORTEST PATHS FOR NETWORK',I4)
C
C
   81 FOUND=1
      CALL MATCH(IFORW,IONE,NDTRAN,NODES,IX,I,MATCH1)
```

```
      IF(MATCH1.EQ.1)GO TO 82
      NDTRAN(NODES) = IFORW
      I = NODES
      NODES=NODES+1
   82 CALL MATCH(IBACK,IONE,NDTRAN,NODES,IX,J,MATCH2)
C
C
      IF(MATCH2.EQ.1)GO TO 88
      NDTRAN(NODES) = IBACK
      J=NODES
      NODES=NODES+1
   88 IMILES = AMILES*10. + 0.5
      IF((ITYPE.EQ.3).OR.(ITYPE.EQ.6)) GO TO 110
   90 DISTAN(I,J) = IMILES
      IF(ITYPE.EQ.2.OR.ITYPE.EQ.5)GO TO 70
  110 DISTAN(J,I) = IMILES
      GO TO 70
C
C
  116 NODES = NODES-1
      DO 10 I=1,5
   10 NETS(IAM) = STORE(HIARCH(I),NETS(IAM),I)
      CALL RANK(IONE,NODES,NDTRAN,INDEXT,INODES)
      DO 40 I=1,NODES
      NDTRAN(I) = INDEXT(I)
      IL=INODES(I)
      DO 40 J=1,NODES
      JL = INODES(J)
   40 IPOLFL(I,J) = DISTAN(IL,JL)
       DO 41 I=1,NODES
      DO 41 J=1,NODES
      DISTAN(I,J) = IPOLFL(I,J)
      IPOLFL(I,J) = 0
      IF(DISTAN(I,J).NE.NINE16)IPOLFL(I,J)=J
   41 CONTINUE
      RETURN
      END
      SUBROUTINE SPATH(NODES)
C**********************************************************************
C  THIS SUBROUTINE USES FLOYD'S ALGORITHM TO FIND THE SHORTEST PATHS
C      FROM ALL NODES TO ALL OTHER NODES (OR FROM ALL NODES TO A
C      SPECIFIC DESTINATION IF J=CONSTANT.)
C      SEE 'AN APPRAISAL OF SOME SHORTEST-PATH ALGORITHMS' BY STUART E.
C      DREYFUS, ORSA-1969, MAY-JUNE, VOLUME 17, NUMBER 3, PG. 395-412
C  VARIABLES :
C  DISTAN  : DISTANCE MATRIX FOR FLOYD'S ALGORITHM
C**********************************************************************
      IMPLICIT INTEGER (D)
      COMMON/IO/IN,IOUT
      COMMON/BLOCK1/NINE16,INFIN,KINFIN,ITWO16,NETS(400)
```

```
      COMMON/BLOCK2/DISTAN(75,75),IPOLFL(75,75)
      INTEGER TEST
C
C
C
      WRITE(IOUT,163)NODES
  163 FORMAT(/' SPATH,  NODES=',I6)
C
C  THE DISTANCE MATRIX DISTAN(I,J) HAS BEEN CREATED IN SUBR. SETUP.
C
C  FLOYD'S ALGORITHM.  K IS THE ITERATION NUMBER.
C
      DO 50 K=1,NODES
      DO 50 I=1,NODES
       IF(I.EQ.K) GO TO 50
      DISTIK = DISTAN(I,K)
      IF(DISTIK.EQ.9999)GO TO 50
      DO 49 J=1,NODES
      IF(I.EQ.J)GO TO 49
      IF(K.EQ.J)GO TO 49
      DISTKJ = DISTAN(K,J)
      IF(DISTKJ.EQ.INFIN)GO TO 49
      TEST = DISTIK+DISTKJ
C
C
      IF(DISTAN(I,J).LE.TEST)GO TO 49
      DISTAN(I,J) = TEST
      IPOLFL(I,J)=IPOLFL(I,K)
   49 CONTINUE
   50 CONTINUE
      RETURN
      END
      SUBROUTINE RANK(NGO,N,R,B,II)
C  R = ARRAY TO BE RANKED.
C  ONLY THAT PART OF R FROM R(NGO) TO R(N) IS RANKED
C  NGO = STARTING SUBSCRIPT OF ARRAY R
C  N = ENDING SUBSCRIPT OF ARRAY R
C  B = RANKED ARRAY.
C  ARRAY II KEEPS TRACK OF THE SUBSCRIPTS OF R
C  II(I) SHOULD BE INITIALIZED IN CALLING PROGRAM WITH II(I)=I, I=1,2,..
C PRIOR TO FIRST CALL OF RANK.
C  REMOVE C IN COLUMN 1 OF STATEMENTS PERTAINING TO II IN ORDER TO INVOK
C  II.
      INTEGER R,B
      DIMENSION R(1),B(1)
C      DIMENSION II(1)
       DIMENSION II(1)
      NG1 = NGO+1
      NM = NGO-1
      B(NGO) = R(NGO)
```

```
      II(NGO) = NGO
      K=NGO
      DO 70 I=NG1,N
      IF(B(K)-R(I)) 10,10,30
   10 B(I) = R(I)
C     II(I) = I
      II(I) = I
   20 K=I
      GO TO 70
   30 DO 50 L=NGO,K
      J=I-L+NM
      IF(B(J)-R(I)) 60,60,40
   40 B(J+1) = B(J)
C      II(J+1) = II(J)
       II(J+1) = II(J)
   50 CONTINUE
      B(J) = R(I)
C     II(J) = I
      II(J) = I
      GO TO 20
   60 B(J+1) = R(I)
C     II(J+1) = I
      II(J+1) = I
      K=I
   70 CONTINUE
      RETURN
      END
      SUBROUTINE CNTLRD(ILAST)
C***********************************************************************
C  THIS SUBROUTINE READS THE CONTROL FILE TO DETERMINE WHICH SHORTEST
C       PATHS SHOULD BE FOUND.
C  VARIABLES:
C  NETNUM(I) : NUMBER OF I-TH NETWORK
C  IFLAG(I) = 0 : FIND ALL SHORTEST PATHS FOR NETWORK I
C           = 1 : FIND ALL PATHS FROM ALL OTHER NODES TO IDEST(I)
C           = 2 : FIND ALL PATHS FROM IORIG(I) TO ALL OTHER NODES
C           = 3 : FIND SHORTEST PATH FROM IORIG(I) TO IDEST(I)
C  IORIG(I)   : ORIGIN OR SOURCE NODE, NETWORK I
C  IDEST(I)   : DESTINATION NODE FOR NETWORK I
C  NETS(I)    : ARRAY TO STORE VALUES OF NETNUM
C  ICNTLS     :  VALUES OF IFLAG
C  ISOURC     :  VALUES OF IORIG
C  ISINK      :  VALUES OF IDEST
C***********************************************************************
      COMMON/IO/IN,IOUT
      COMMON/BLOCK1/NINE16,INFIN,KINFIN, ITWO16,NETS(400)
      DIMENSION NETNUM(7)
      ILAST = 0
   30 READ(IN,40,END=9999)(NETNUM(I),IFLAG,IORIG,IDEST,I=1,7)
      WRITE(IOUT,40)(NETNUM(I),IFLAG,IORIG,IDEST,I=1,7)
```

```
   40 FORMAT(7(I3,I1,2I3))
      DO 50 I=1,7
      IF(NETNUM(I).LE.0) GO TO 9999
      ILAST=ILAST+1
   50 NETS(ILAST) = NETNUM(I)
C
      GO TO 30
 9999 REWIND IN
      RETURN
      END
      SUBROUTINE PRNOUT(DISTAN,NODES,IAM)
      COMMON/IO/IN,IOUT
      COMMON/BLOCK1/NINE16,INFIN,KINFIN, ITWO16, NETS(400)
      COMMON/BLOCK2/IDTAN(75,75),FILET(75,75),NOTRAN(75),INDEXT(75)
      INTEGER DISTAN(75,75), HIARCH(5)
C
C  PRINT OUT FOR FLOYD'S ALGORITHM
C
      DO 1 I=1,5
    1 HIARCH(I)=UNMASK(NETS(IAM),I)
      NETID = HIARCH(5)
      WRITE(IOUT,10)HIARCH
   10 FORMAT(/'            MATRIX FOR NETWORK :  ZONE =',I5/34X'EOC =',I5
     1,30X'GROUP =',I5/31X'SECTOR =',I5/31X'UNIT AREA =',I5/)
      KGO=1
   23 KWIT=KGO+14
      IF(KWIT.GT.NODES)KWIT=NODES
      WRITE(IOUT,26)
      WRITE(IOUT,26)(NOTRAN(K),K=KGO,KWIT)
      DO 24 J=1,NODES
      WRITE(IOUT,28)NOTRAN(J),(DISTAN(J,K),K=KGO,KWIT),NOTRAN(J)
   24 CONTINUE
      WRITE(IOUT,26)(NOTRAN(K),K=KGO,KWIT)
      KGO = KWIT+1
      IF(KGO.LE.NODES)GO TO 23
   26 FORMAT(6X,15I8)
   28 FORMAT(2XI4,15I8,I4)
C
      RETURN
      END
      SUBROUTINE MATCH(A,M,B,N,II,JJ,IMATCH)
C
C  SUBR. MATCH COMPARES ELEMENTS OF INTEGER ARRAYS A AND B FOR
C  EQUALITY.  EQUAL ELEMENTS ARE A MATCH.  RETURNED VALUES ARE:
C     IMATCH (= 0 = NO MATCH; = 1 = A MATCH)
C     II & JJ, THE INDICES OF THE MATCHED ELEMENTS IN A AND B RESPECTIVE
C
      INTEGER A(M), B(N)
      IMATCH=1
      DO 1 I=1,M
```

```
      II=I
      DO 1 J=1,N
      JJ=J
      IF(A(I).EQ.B(J))RETURN
    1 CONTINUE
      IMATCH = 0
      RETURN
      END
      FUNCTION STORE(VALUE,WORD,I)
      IMPLICIT INTEGER (A-Z)
      DIMENSION MASK(5), POWER(5)
      LOGICAL STRIP(5),P,LWORD
      DATA FIRST/0/
      IF(FIRST.EQ.1)GO TO 10
C
C  CREATE MASKS (ALL BITS ON) IN 5 DIFFERENT BIT FIELDS TO BE USED FOR
C  STORING OR EXTRACTING CORRESPONDING BIT FIELDS IN A WORD.  BEGINNING
C  WITH THE HIGHEST ORDER BITS THE NUMBER OF BITS IN EACH FIELD ARE:
C  5, 6, 6, 6, 9.  THESE FIELDS CORRESPOND TO VALUES IN THE ARRAY HIARCH
C  WHICH ARE ZONE I.D., EOC I.D., GROUP I.D., SECTOR I.D. AND NETWORK
C  (OR, UNIT AREA) I.D.  THESE 5 VALUES ARE STORED/EXTRACTED IN WORD.
C
    2 FIRST = 1
      POWER(1) = 2**27
      POWER(2) = 2**21
      POWER(3) = 2**15
      POWER(4) = 2**6
      POWER(5) = 1
       TWO5 = 2**5-1
      MASK(1) = TWO5*POWER(1)
      TWO6 = 2**6 - 1
      MASK(2) = TWO6*POWER(2)
      MASK(3) = TWO6*POWER(3)
      MASK(4) = TWO6*POWER(4)
      MASK(5) = 2**9-1
      DO 1 J=1,5
    1 STRIP(J) = MASK(J)
C
C  STORE VALUE IN THE I-TH BIT FIELD OF WORD.
C
   10 LWORD = WORD
      P=.NOT.STRIP(I)
      LWORD = LWORD.AND.P
      K=VALUE*POWER(I)
      P = K
      P = LWORD.AND.P
      STORE = P
      RETURN
      ENTRY UNMASK(WORD,I)
      LWORD=WORD
```

```
P = LWORD.AND.STRIP(I)
UNMASK = P
UNMASK = UNMASK/POWER(I)
RETURN
END
```

```
C     SUBROUTINE ALLNET
C  REMOVE THE ABOVE 'C' TO USE ALLNET AS A SUBROUTINE CALLED BY NETWORK.
C  ALSO REPLACE 'STOP' WITH 'RETURN'.
C
C
C  RTI.C43.P04956.JWD.ALLNET.FORT
C
C     THE OBJECT OF THIS STEP IS TO CREATE THE INITIAL DISTANCE MATRIX
C  OF DIRECTLY LINKED UNIT AREAS AS 2 MATRICES STORED BY ROWS ON OUTPUT
C  DEVICES 91 AND 92.  THE MATRICES ARE INPUT TO THE NEXT STEP, DAMAGE
C  ASSESSMENT, WHICH ADJUSTS THE DISTANCES PER 'BOS' CODE VALUES AND
C  ADDS THE  CORRESPONDING ELEMENTS IN EACH ROW.
C     FILE 53 CONTAINS RESULTS OF SHORTEST PATHS COMPUTED WITHIN EACH
C  UNIT AREA, I.E. NETWORK, AND THE IDENTIFICATION OF THE NODES IN EACH
C  NETWORK.
C     TWO UNIT AREAS A AND B ARE LINKED IF THEY HAVE ONE OR MORE BOUNDAR
C  NODES IN COMMON.
C     LINKAGE OF A AND B IS DETERMINED BY COMPARISON OF THE LIST OF
C  NODES OF EACH NETWORK.  EQUAL NODE I.D.'S OCCUR FOR COMMON NODES
C  ON THE BONDARY BETWEEN THE TWO NETWORKS.
C     IF NETWORKS A AND B ARE DIRECTLY LINKED, THE DISTANCE FROM A TO B
C  IS DEFINED AS THE AVERAGE OVER SHORTEST PATHS FROM ALL NODES IN A TO
C  THE COMMON BOUNDARY NODES PLUS THE AVERAGE OVER SHORTEST PATHS FROM
C  THE COMMON BOUNDARY NODES TO ALL NODES IN B.
C
      DIMENSION NETS(400),NODINA(75),NODINB(75),SPATHA(75,75),SPATHB(75,
     1 75)
      INTEGER A2BOUN(400),B2BOUN(400),BOUN2A(400),BOUN2B(400),
     1 TOBOUN/91/, FRMBOU/92/, SPATHA,SPATHB
      COMMON/SHIFT/ITWO16,INFIN,A2BOUN,B2BOUN,BOUN2A,BOUN2B
C
      ITWO16 = 2**16
      KINFIN = 9999
       INFIN = KINFIN*ITWO16
      READ(30)MAXNET,NETS
      REWIND 30
      NUMNET = MAXNET
C
C  I/O DEVICES:
      I80 = 80
      I70 = 70
      I71 = 71
      I81 = 81
C
      DO 10 N=1,400
      A2BOUN(N)=INFIN
      BOUN2B(N) = INFIN
      B2BOUN(N)=INFIN
   10 BOUN2A(N) = INFIN
C
```

7-2 Listing of Fortran Source Program ALLNET

```
C     INITIALIZE FILES 70 AND 80;  ASSUMPTION IS THAT ALL DIRECTLY LINKED
C  DISTANCES INITIALLY EQUAL INFINITY (=9999).
C
      DO 11 N=1,400
      WRITE(I70)N,(INFIN,I=1,400)
   11 WRITE(I80)N,(INFIN,I=1,400)
      REWIND I70
      REWIND I80
C
C     INITIALIZE LOOP INDEX TO COUNT UNIT AREA 'A'
C
      IA = 0
    1 IA = IA+1
      IF(IA.GT.MAXNET) GO TO 200
C
C     INITIALIZE THE VECTOR OF DIRECTLY LINKED DISTANCES FOR NETWORK A
C  THE VECTOR IS THE IA-TH. ROW OF THE DISTANCE MATRIX
C
      READ(I70,END=100)INDEXA,A2BOUN
      READ(I80) INDEXA,BOUN2B
C
C     READ FILE 53 FOR THE RESULTS OF THE SHORTEST PATHS COMPUTATION FOR
C  UNIT AREA A.  THE FIRST RECORD CONTAINS NO. OF NETWORKS, AND NET ID'S
C
      READ(53) INDEXA,NRNODA,NODINA,SPATHA
      NETNOA = UNMASK(INDEXA,5)
      A2BOUN(IA) = NETNOA
      BOUN2B(IA) = NETNOA
      MAXA = NODINA(NRNODA)
C     PRINT 1492,IA,NETNOA,NRNODA,(NODINA(LM),LM=1,NRNODA)
 1492 FORMAT(30I4,2(/8X30I4))
C
C     NETWORK A WILL BE COMPARED WITH NETWORKS B (B=IA+1,IA+2,....,
C ....,# NETWORKS).
C
      IB=IA
    2 IB=IB+1
      IF(IB.GT.MAXNET)GO TO 100
      READ(I70,END=100)INDEXB,B2BOUN
      READ(I80)INDEXB,BOUN2A
C
C  ASSUME INITIALLY THAT A AND B ARE NOT LINKED AND THE DISTANCE A TO B
C  IS INFINITE.
C
      B2BOUN(IA) = INFIN
      BOUN2A(IA) = INFIN
C
C  READ THE SHORTEST PATHS RESULTS FOR NETWORK B
C
      READ(53,END=100) INDEXB,NRNODB,NODINB,SPATHB
```

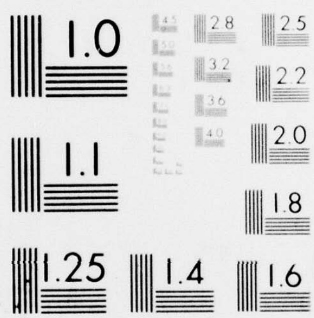
```
      NETNOB = UNMASK(INDEXB,5)
      B2BOUN(IB) = NETNOB
      BOUN2A(IB) = NETNOB
C     PRINT 1492,IB,NETNOB,NRNODB,(NODINB(LM),LM=1,NRNODB)
C
C  L WILL BE USED TO COUNT THE NUMBER OF MATCHED NODES IN A AND B.
C
      L=0
      A2BOU = 0.
      B2BOU = 0.
      BOU2B = 0.
      BOU2A = 0.
      JGO=1
      MAXB = NODINB(NRNODB)
      DO 4 I=1,NRNODA
      ITHNOD = NODINA(I)
C     PRINT 1492,I,ITHNOD
      IF(ITHNOD.LT.NODINB(JGO))GO TO 4
      DO 3 J=JGO,NRNODB
      JTHNOD = NODINB(J)
     .IF(ITHNOD.GT.MAXB.OR.JTHNOD.GT.MAXA)GOTO 45
C     PRINT 1493,J,JTHNOD
 1493 FORMAT(5X2I5,' J, JTHNOD')
      IF(JTHNOD.GT.ITHNOD)JGO=J
      IF(JTHNOD.GT.ITHNOD)GO TO 4
      IF(JTHNOD.LT.ITHNOD)GO TO 3
C
C  A MATCH BETWEEN ITH NODE IN A AND J-TH NODE IN B
C
      L = L+1
C  THE FOLLOWING GIVE DIST A TO B AS ( A TO BOUND. + BOUND. TO B).
      A2BOU = A2BOU+COLSUM(SPATHA,NRNODA,I)
      BOU2B = BOU2B + ROWSUM(SPATHB,NRNODB,J)
C  THE FOLLOWING GIVE DIST B TO A AS (B TO BOUND. + BOUND. TO A).
      B2BOU = B2BOU + COLSUM(SPATHB,NRNODB,J)
      BOU2A = BOU2A + ROWSUM(SPATHA,NRNODA,I)
      JGO = JGO+1
      GO TO 4
    3 CONTINUE
    4 CONTINUE
C  L = 0 = NO NODES IN COMMON BETWEEN NETWORKS A AND B.
   45 IF(L.EQ.0)GO TO 46
      IA2BOU = A2BOU/(L*NRNODA) + 0.5
      A2BOUN(IB) = IA2BOU*ITWO16 + NETNOB
      IBOU2B = BOU2B/(L*NRNODB) + 0.5
      BOUN2B(IB) = IBOU2B*ITWO16 + NETNOB
C
C  THIS COMPLETES THE IB-TH. ROW OF ARRAY 'TO BOUN.' UP TO THE IATH.
C  ELEMENT:
C
```

AD-A038 265 RESEARCH TRIANGLE INST RESEARCH TRIANGLE PARK N C OPE--ETC F/G 15/3
LOCAL EMERGENCY OPERATING SYSTEM - LEMOS.(U)
JUL 76 J W DUNN, R N HENDRY, R O LYDAY DAHC20-73-C-0253

UNCLASSIFIED RTI-44U-873 NL

2 OF 2
AD
A038265

END
DATE
FILMED
5-77



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
      IB2BOU = B2BOU/(L*NRNODB) + 0.5
      B2BOUN(IA) = IB2BOU*ITWO16 + NETNOA
C
C  THIS COMPLETES THE IB-TH. ROW OF ARRAY 'FROM BOUND.' UP TO THE IATH.
C  ELEMENT:
C
      IBOU2A = BOU2A/(L*NRNODA) + 0.5
      BOUN2A(IA) = IBOU2A*ITWO16 + NETNOA
   46       INDEXB = NETNOB*ITWO16 + IB
      WRITE(I71) INDEXB,B2BOUN
      WRITE(I81) INDEXB,BOUN2A
C
C  CONTINUE PROCESSING UNIT AREA A AGAINST THE NEXT UNIT AREA B.
C
      GO TO 2
C
C  100 = HAVE PROCESSED ALL B NETWORKS AGAINST NETWORK A.
C  WRITE THE RESULTS, A2BOUN(J) AND BOUN2B(J),J=1,2...,#NETWORKS,
C  ON FILES 'TOBOUN' AND 'FRMBOU'.
C
  100 REWIND 53
      REWIND 70
      REWIND 80
      REWIND 71
      REWIND 81
      INDEXA = NETNOA*ITWO16 + IA
      WRITE(TOBOUN) INDEXA,A2BOUN
      WRITE(FRMBOU)INDEXA,BOUN2B
C
C     ADVANCE FILE 53 TO THE NEXT UNIT AREA A WHICH IS TO BE PROCESSED
C  AGAINST ALL OTHER B UNIT AREAS EXCEPT B<A.
C
      DO 12 L=1,IA
   12 READ(53,END=200)
C
C     PART OF THE NEXT VECTOR A HAS ALREADY BEEN PROCESSED AS A VECTOR
C  B UP TO THE IA-TH. ELEMENT AND THE RESULTS WERE WRITTEN ON FILES I71
C  AND I81
C  SO READ THOSE FILES FOR VECTOR A INSTEAD OF READING I70 AND I80.
C  TO DO THIS, FLIP-FLOP THE VALUES FOR I80 AND I70, I81 AND I71.
      I70 = 141-I70
      I71 = 141-I71
      I80 = 161-I80
      I81 = 161-I81
      GO TO 1
C
C  200 = ALL UNIT AREA NETWORKS HAVE BEEN PROCESSED AS NETWORK 'A'
C  FOR 'A' = 1, 2, ... , IA-1.
C
  200 REWIND 53
```

```
      REWIND 70
      REWIND 80
      REWIND 71
      REWIND 81
      REWIND TOBOUN
      REWIND FRMBOU
C
C     THE INITIAL DISTANCE MATRIX OF DIRECTLY LINKED NODES, E.G. UNIT
C  AREAS, NOW EXISTS AS 2 FILES OF ROWS ON DEVICES 91 AND 92.  THE FILES
C  ARE DISTANCES FROM ALL NODES IN A TO THE BOUNDARY BETWEEN A AND B
C  AND THE DISTANCES FROM THE BOUNDARY BETWEEN A AND B TO ALL NODES IN B
C
C
C  USING ARRAY BOUN2A FOR WORKING STORAGE IN SUBR. NETPRN:
C
      CALL NETPRN(MAXNET,BOUN2A,TOBOUN,1)
      CALL NETPRN(MAXNET,BOUN2A,FRMBOU,1)
      CALL NETPRN(MAXNET,BOUN2A,TOBOUN,2)
      CALL NETPRN(MAXNET,BOUN2A,FRMBOU,2)
      STOP
C
C  TO USE ALLNET AS A SUBROUTINE, REMOVE 'STOP', AND REMOVE 'C' IN
C 'RETURN' CARD:
C     RETURN
C
      END
      FUNCTION ROWSUM(DISTAN,NODES,ROWNUM)
C
C  THESE 2 FUNCTIONS ARE APPLIED TO THE DISTANCE MATRICES COMPUTED FOR
C  THE INDIVIDUAL UNIT AREA NETWORKS.
C
C
C  COMPUTE THE SUM OF DISTANCES FROM NODE ROWNUM TO ALL NODES.
C
      INTEGER DISTAN(75,75),ROWNUM,COLNUM
      ROWSUM = 0.
      DO 1 J=1,NODES
    1 ROWSUM = ROWSUM + DISTAN(ROWNUM,J)
      RETURN
      ENTRY COLSUM(DISTAN,NODES,COLNUM)
C
C  COMPUTE THE SUM OF DISTANCES FROM ALL NODES TO NODE COLNUM.
C
      COLSUM = 0.
      DO 2 I=1,NODES
    2 COLSUM = COLSUM + DISTAN(I,COLNUM)
      RETURN
      END
      SUBROUTINE NETPRN(N,LINEIN,IO,IWHICH)
C
```

```
C  IWHICH = 1 = PRINT OUT DISTANCE VALUES - UPPER HALF OF WORD
C  IWHICH = 2 = PRINT OUT POLICY VALUES - LOWER HALF OF WORD
C
      DIMENSION LINEIN(1),LINOUT(31)
      COMMON/SHIFT/ITWO16,INFIN
C
C  OUTPUT DEVICE IOUT:
      IOUT = 3
      JGO=1
    5 WRITE(IOUT,3)
    3 FORMAT(1H1)
    1 READ(IO,END=100)INDEX,(LINEIN(L),L=1,N)
      JKWIT = JGO+30
      IF(JKWIT.GT.N)JKWIT=N
      LL=0
      DO 4 L=JGO,N
      LL = LL+1
      LINOUT(LL)=LINEIN(L)/ITWO16
      IF(IWHICH.EQ.2)LINOUT(LL)=LINEIN(L) - LINOUT(LL)*ITWO16
    4 CONTINUE
      INDEX1 = UNMASK(INDEX,5)
      WRITE(IOUT,2)INDEX1,(LINOUT(L),L=1,LL)
    2 FORMAT(1XI3,31I4)
      GO TO 1
  100 REWIND IO
      JGO = JKWIT+1
      IF(JGO.GT.N)RETURN
      GO TO 5
      END
```

```
C
C  TO USE 'DAMAGE' AS A SUBROUTINE, REMOVE 'C' IN SUBROUTINE CARD AND
C  CHANGE 'STOP' TO 'RETURN'.
C
C     SUBROUTINE DAMAGE
C  RTI.C41.P04956.JWD.DAMAGE.FORT
C  PERFORMS DAMAGE ASSESSMENT OF UNIT AREAS.
C  ASSIGN NEW DISTANCES USING BOS VALUE ON RESOURCE FILE
C  THE RESOURCE FILE IS ON FT61F001.
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION NETS(400),DAMAGE(400),A2BOUN(400),BOUN2B(400),
     1 DISTAN(400)
      COMMON/SHIFT/ITW016,INFIN
      ITW016 = 2**16
      KINFIN=9999
      INFIN = KINFIN*ITW016
      READ(30)MAXNET,NETS
      REWIND 30
      NUMNET = MAXNET
      DO 10 I=1,NUMNET
      NETNO = UNMASK(NETS(I),5)
      IF(NETNO.LE.0)GO TO 20
    8 READ(61,11,END=20) ZONEID,UNITID,LUC,BOS
   11 FORMAT(2I3,I2,3XI1)
      IF(LUC.NE.5)GO TO 8
C
C  FIND A MATCH BETWEEN THE UNIT AREA ID, UNITID, AND SOME NETWORK,
C NETNO, IN THE LINKS FILE.
C
      IF(NETNO.LT.UNITID)GO TO 8
C
C  REDUNDANCY; ASSUMED SORT ON NETNO AND UNITID SHOULD PRECLUDE
C  NETNO>UNITID
C
      IF(NETNO.GT.UNITID)GO TO 10
      DAMAGE(I) = 2**(BOS-1)
   10 CONTINUE
C
C  READ A2BOUN AND BOUN2B FILES AND COMPUTE
C  D(I,J)=A2BOUN(J)*DAMAGE(I) + BOUN2B(J)*DAMAGE(J)
C
   20 REWIND 61
      DO 55 I=1,NUMNET
      READ(91)INDEXA,A2BOUN
      READ(92)INDEXA,BOUN2B
      DO 54 J=1,NUMNET
      IA2BOU = A2BOUN(J)/ITW016
      DISTAN(J) = INFIN
      IF(IA2BOU.EQ.KINFIN)GO TO 54
```

7-3 Listing of Fortran Source Program DAMAGE

```
      IBOU28 = BOUN28(J)/ITWO16
      POLIJ = UNMASK(NETS(J),5)
      DISTAN(J) = (IA28OU*DAMAGE(I) + IBOU28*DAMAGE(J))*ITWO16+POLIJ
   54 CONTINUE
   55 WRITE(93)INDEXA,DISTAN
      REWIND 91
      REWIND 92
      REWIND 93
      CALL NETPRN(MAXNET,A28OUN,93,1)
      CALL NETPRN(MAXNET,A28OUN,93,2)
      STOP
C
C     RETURN
C
      END
```

```
C   RTI.C43.PU4956.JWD.FLOYD.FORT
C
C   TO USE 'FLOYD' AS A SUBROUTINE, REMOVE C IN SUBROUTINE CARD AND
C   CHANGE 'STOP' TO 'RETURN'
C
C      SUBROUTINE FLOYD
C
C   PROGRAM FLOYD.  COMPUTES SHORTEST PATHS IN DIST(I,J) USING
C   FLOYD'S ALGORITHM ADAPTED TO WORK WITH ONLY 2 ROWS OF D
C   IN CORE AT ONE TIME.  D(I,J) IS STORED BY ROWS ON DEVICE 93.
C   THE SOLUTION MATRIX IS WRITTEN TO DEVICE 55.
C
       INTEGER USING(400),CHANGE(400),TEST,DISTIK,DISTIJ,DISTKJ
       ITWO16 = 2**16
       KINFIN = 9999
       INFIN = KINFIN*ITWO16
       I70 = 70
       I80 = 80
       NEXUSE = 60
       KGO=1
       READ (30) NUMNET,NETS
       REWIND 30
    39 DO 12 I=1,NUMNET
       READ(93)INDEXI,CHANGE
    12 WRITE(I70)INDEXI,CHANGE
       REWIND 93
       REWIND I70
C
C   INITIALIZE THE FIRST 'USING' ROW
C
       READ(I70)INDEXI,USING
       REWIND I70
       WRITE(NEXUSE)INDEXI,USING
       REWIND NEXUSE
C
C   BEGIN FLOYD'S ALGORITHM.
C
       DO 40 K=KGO,NUMNET
C
C   EVALUATE ALL CURRENT SHORTEST PATHS D(I,J) USING EACH NODE K
C   IN TURN AS AN INTERMEDIATE NODE IN THE PATH.  IF THE DISTANCE USING
C   NODE K IS LESS THAN THE DISTANCE WITHOUT, INSERT NODE K INTO THE
C   PATH.  D(I,J) IS STORED BY ROWS ON DEVICES 93 AND I70.
C
C
C   READ IN THE 'USING-TO-CHANGE' ROW, I.E. THE K-TH ROW:
C
       READ(NEXUSE)INDEXK,USING
       REWIND NEXUSE
       DO 30 I=1,NUMNET
```

7-4 Listing of Fortran Source Program FLOYD

```
C
C     READ IN THE 'TO-BE-CHANGED' ROW, I.E. THE I-TH. ROW
C
         READ(I70)INDEXI,CHANGE
         IF(K.EQ.I)GO TO 21
C
C     DISTANCE I TO K IS STORED AS AN INTEGER IN UPPER HALF OF THE WORD
C     CHANGE(K):
C
         DISTIK = CHANGE(K)/ITWO16
          IF(DISTIK.EQ.KINFIN)GO TO 21
C
C     UPDATE THE J = 1,2,...,NUMNET ELEMENTS IN THE ROW, I, BEING
C     CHANGED.
C
         DO 20 J=1,NUMNET
         IF(K.EQ.J)GO TO 20
         IF(I.EQ.J)GO TO 20
         DISTKJ = USING(J)/ITWO16
         IF(DISTKJ.EQ.KINFIN)GO TO 20
         TEST = DISTIK + DISTKJ
         IF(TEST.LT.CHANGE(J)/ITWO16)CHANGE(J)=TEST*ITWO16+(CHANGE(K)
        1 - DISTIK*ITWO16)
C
C     STORE THE NEW DISTANCE IN THE UPPER HALF OF WORD CHANGE(J) AND
C     STORE THE NEW POLICY IN THE LOWER HALF.
C
      20 CONTINUE
C
C     WRITE THE CHANGED ROW ON DEVICE I80:
C
      21 WRITE(I80)INDEXI,CHANGE
C
C     SAVE THE SOLUTION MATRIX ON FILE 55
C
         IF(K.EQ.NUMNET)WRITE(55)INDEXI,CHANGE
         IF(I.NE.K)GO TO 29
         DO 28 II=1,NUMNET
      28 USING(II) = CHANGE(II)
      29 IF(I.NE.K+1)GO TO 30
         WRITE(NEXUSE)INDEXI,CHANGE
         REWIND NEXUSE
      30 CONTINUE
C
C     FLIP-FLOP THE I/O DEVICES:
C
         I70 = 150 - I70
         I80 = 150 - I80
         REWIND 70
         REWIND 80
```

```
   40 CONTINUE
      REWIND 55
C
C  END OF FLOYD'S ALGORITHM.
C
      STOP
C
C  TO USE AS A SUBROUTINE, REMOVE 'STOP' AND 'C' IN RETURN CARD.
C
C     RETURN
C
      END
```

```
C  RTI.C43.P04956.JWD.TVLREC.FORT
C
C  TO USE TVLREC AS A SUBROUTINE, REMOVE 'C' IN SUBROUTINE CARD AND
C  REPLACE 'STOP' WITH 'RETURN'.
C
C     SUBROUTINE TVLREC
C
C  PROGRAM TVLREC:  WRITES RECORDS FOR TVLREC FILE ON DEVICE 94.
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION NETS(400),DISTAN(400)
      ITWO16 = 2**16
C
C  ASSUMED VELOCITY OF TRAVEL, SPEED, IS 45 MILES/HOUR.
C
      SPEED = 45
      READ(30)NUMNET,NETS
      REWIND 30
C
C  TRAVEL TIMES ARE TRAVEL TIMES BETWEEN UNIT AREAS.
C  THEREFORE THE ''LEVEL OF DATA'' IS 5 CORRESPONDING TO LOWEST LEVEL.
C
      LVLDAT = 5
      DO 50 I=1,NUMNET
      NETNUM = UNMASK(NETS(I),5)
   30 READ(55)INDEXI,DISTAN
      ORIGIN = UNMASK(INDEXI,5)
      IF(ORIGIN.NE.NETNUM)GO TO 30
      TVLCOD = 2
      REFZON = UNMASK(NETS(I),1)
C
C  EDC I.D. IS UNMASK(NETS(I),2) - NOT REQUIRED IN TVLREC RECORDS.
C
      RGROUP = UNMASK(NETS(I),3)
      RSECTR = UNMASK(NETS(I),4)
      WRITE(94,9)REFZON,RGROUP,RSECTR,LVLDAT,REFZON,ORIGIN,TVLCOD
    9 FORMAT(3I2,I1,4X,2I3,I1)
      TVLCOD = 3
      DO 49 J=1,NUMNET
      IF(I.EQ.J)GO TO 49
      DISTIJ = DISTAN(J)/ITWO16
C
C  DISTANCES ARE IN MILES*10, I.E. DISTIJ = 26 = 2.6 MILES, FOR EXAMPLE.
C
      TVLTIM = (DISTIJ*10)/SPEED
C
C  ZONE IS ZONE OF THE DESTINATION AREA.
C
      ZONE = UNMASK(NETS(J),1)
C
```

7-5 Listing of Fortran Source Program TVLREC

```
C  DEST IS THE IDENTIFICATION OF THE DESTINATION AREA.
C
      DEST = UNMASK(NETS(J),5)
      WRITE(94,10)REFZON,RGROUP,RSECTR,LVLDAT,TVLTIM,ZONE,DEST,TVLCOD
   49 CONTINUE
   10 FORMAT(3I2,I1,I4,2I3,I1)
   50 CONTINUE
      STOP
C
C  TO USE AS A SUBROUTINE, REPLACE 'STOP' WITH 'RETURN'
C
C     RETURN
C
      END
```